EuroCG

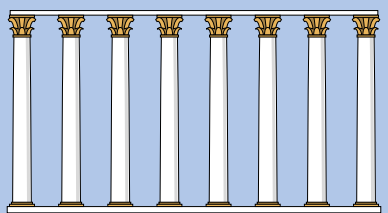
St. Petersburg

2021 April 7-9 2021

Book of Abstracts

37th European Workshop on Computational Geometry

April 7–9, 2021 in St. Petersburg, Russia



Preface

Originally, the 37th European Workshop on Computational Geometry (EuroCG 2021) was scheduled to be held on April 7-9, 2021 at Saint Petersburg State University, Russia. EuroCG is an annual workshop that combines a strong scientific tradition with a friendly and informal atmosphere. Traditionally, the workshop is a forum where researchers can meet, discuss their work, present their results, and establish scientific collaborations, in order to promote research in the field of Computational Geometry, within Europe and beyond.

Due to the spread of the COVID-19, and due to the persistence of the pandemic and travel restrictions, in December 2020 we took the decision to organize EuroCG 2021 as a completely online event. To preserve the tradition of an informal gathering and to foster spontaneous communication among the participants, we have selected Gather Town (`https://gather.town`) as a virtual conference venue. The space designed specifically for EuroCG 2021 contained two rooms for the two parallel sessions; the lecture hall for the introductory meeting, business meeting and invited talks; and the main lobby, where additionally to the socialisation, the participants could observe the posters announcing the main results of the accepted papers and to continue discussions of the research during the coffee breaks. Additionally we had set up several Discord channels for reporting technical problems and for the discussion of the issues raised during to the business event. All talks were recorded and made available to the registered participants in a private YouTube channel. The overall online event went smoothly with only two talks being rescheduled due to technical problems. Overall EuroCG 2021 had 262 registered participants, which is a double of the usual number.

We received 88 submissions, which underwent a limited refereeing process by the program committee in order to ensure that acceptance policy is met. Finally 72 submissions were selected for presentation at the workshop. The selected papers were split into 18 sessions which were spread over three days of the workshop. Each paper was supported by a poster that appeared at a poster stand in the main lobby for 20 minutes following the session in which it was presented.

To encourage students to present their papers and to honour the outstanding presentations, we announced a Best Student Presentation Award. The winners were decided by voting among the workshop participants. In total, 77 votes were cast. The best presentation award was given to Sören Nickel (15 votes) for presenting the paper "Recognition of Unit Disk Graphs for Caterpillars, Embedded Trees, and Outerplanar Graphs" authored by Sujoy Bhore, Sören Nickel and Martin Nöllenburg. The second place was given to Leonie Ryvkin (13 votes) for presenting the paper "On the Realizability of Free Space Diagrams" authored by Maike Buchin, Leonie Ryvkin and Carola Wenk. The third place was given to Eva-Maria Hainzl (12 votes) for presenting the paper: "Geometric Dominating Sets – A Minimum Version of the No-Three-In-Line Problem", authored by Oswin Aichholzer, David Eppstein and Eva-Maria Hainzl. Congratulations to the young scientists for their successful efforts to clearly convey to the audience their scientific results!

In addition to the 72 contributed talks, this book contains abstracts of the invited lectures and two tutorials. The invited speakers were Anna Lubiw, János Pach, Gaiane Panina. They were watched online by between 118 and 186 people. The tutorials were given by Serguei Barannikov and Manfred Scheucher and took place on Friday, as the last events of the workshop.

During the business meeting, Emilio Di Giacomo and Fabrizio Montecchiani presented Perugia, where EuroCG 2022 will take place during March 14-16. There was a single bid for organizing EuroCG 2022 by Rodrigo Silveira, who together with Clemens Huemer, Carlos Seara, and David Orden will organize EuroCG 2022 in Barcelona.

We gratefully thank all authors, speakers, invited speakers, program committee, external reviewers, session chairs, local organizing team and participants for their contribution to the success of this event. We gratefully thank Leonhard Euler International Mathematical Institute in Saint Petersburg (EIMI), St. Petersburg Department of Steklov Mathematical Institute of Russian Academy of Sciences (PDMI RAS), and St. Petersburg State University (SPbU) for making this event possible and for helping us to make the participation in EuroCG 2021 free of charge for all the participants. The workshop is financially supported by a grant from the Government of the Russian Federation, agreements 075-15-2019-1619 and 075-15-2019-1620 and by a grant from Simons Foundation.

EuroCG does not have formally published proceedings; therefore, we expect most of the results outlined here to be also submitted to peer-reviewed conferences and/or journals. This book of abstracts, available through the EuroCG 2021 web site, should be regarded as a collection of preprints.

We are hoping to see you **in person** in Perugia in April next year!

Saint Petersburg, April 20

Elena Arseneva
Tamara Mchedlidze
(EuroCG 2021 co-chairs)

Local Organisation Committee

Elena Arseneva (co-chair)
Tamara Mchedlidze (co-chair)
Boris Zolotov
Asya Gilmanova

Funded by:

Leonhard Euler
International Mathematical Institute
in Saint Petersburg

St Petersburg
University

St. Petersburg Department
of Steklov Mathematical Institute
of Russian Academy of Sciences

Program Committee

Elena Arseneva (co-chair)	St. Petersburg University (SPbU)
Gill Barequet	Technion — Israel Institute of Technology
Maike Buchin	Ruhr Universität Bochum
Pilar Cano	Universite libre de Bruxelles
Erin Chambers	Saint Louis University
Jean-Lou De Carufel	University of Ottawa
Ruy Fabila-Monroy	Departamento de Matemáticas, Cinvestav
Michael Hoffmann	ETH Zurich
Matthew Katz	Ben-Gurion University
Deok-Soo Kim	Hanyang University
Linda Kleist	TU Braunschweig
Grigorios Koumoutsos	Universite libre de Bruxelles
Tamara Mchedlidze (co-chair)	Utrecht University
Piotr Micek	Jagiellonian University
Debajyoti Mondal	University of Saskatchewan
Wolfgang Mulzer	Freie Universität Berlin
Martin Nöllenburg	Vienna University of Technology
Evanthia Papadopoulou	University of Lugano (USI)
Irene Parada	TU Eindhoven
Hugo Parlier	University of Luxembourg
Valentin Polishchuk	Linkoping University
André Schulz	FernUniversität in Hagen
Hang Si	Weierstrass Institute for Applied Analysis and Stochastics
Rodrigo Silveira	Universitat Politècnica de Catalunya
Marc Van Kreveld	Utrecht University
Birgit Vogtenhuber	Graz University of Technology
Carola Wenk	Tulane University
André van Renssen	The University of Sydney

Table of Contents

Keeping your Distance: Algorithms and Hardness (Invited Talk) ..... A
Anna Lubiw

Crossing Lemmas for Multigraphs (Invited Talk) ..... B
János Pach

An universality theorem for stressable graphs in the plane (Invited Talk) C
Gaiane Panina

Canonical Forms = Persistence Diagrams (Tutorial) ..... D
Serguei Barannikov

Using SAT Solvers in Combinatorics, Combinatorial Geometry, and Graph Drawing (Tutorial) ..... E
Manfred Scheucher

Many Order Types on Integer Grids of Polynomial Size 1
Manfred Scheucher

Tight bounds on the expected number of holes in random point sets 2
Martin Balko, Manfred Scheucher and Pavel Valtr

Obstructing Classification via Projection 3
Pantea Haghighatkhah, Wouter Meulemans, Bettina Speckmann, Jérôme Urhausen and Kevin Verbeek

Route Reconstruction from Traffic Flow via Representative Trajectories 4
Bram Custers, Wouter Meulemans, Bettina Speckmann and Kevin Verbeek

Route-preserving Road Network Generalization 5
Mees van de Kerkhof, Marc Van Kreveld, Irina Kostitsyna, Maarten Löffler and Tim Ophelders

Path-Greedy Spanner in Near-Quadratic Time: Simpler and Better 6
Paz Carmi and Idan Tomer

Lions and contamination, triangular grids, and Cheeger constants 7
Henry Adams, Leah Gibson and Jack Pfaffinger

A Geometric Approach to Inelastic Collapse 8
Bernard Chazelle, Kritkorn Karntikoon and Yufei Zheng

Max-Min 3-dispersion on a Convex Polygon 9
Yasuaki Kobayashi, Shin-Ichi Nakano, Kei Uchizawa, Takeaki Uno, Yutaro Yamaguchi and Katsuhisa Yamanaka

The maximal number of 3-term arithmetic progressions in finite sets in different geometries 10
Itai Benjamini and Shoni Gilboa

Notes on pivot pairings 11
Barbara Giunti

Improved Bounds for Half-Guarding Monotone Polygons 12
Hannah Miller Hillberg, Erik Krohn and Alex Pahlow

Mapping Multiple Regions to the Grid with Bounded Hausdorff Distance 13
Ivor van der Hoog, Mees van de Kerkhof, Marc Van Kreveld, Maarten Löffler, Frank Staals, Jérôme Urhausen and Jordi L. Vermeulen

A Dynamic Data Structure for k-Nearest Neighbors Queries 14
Sarita de Berg and Frank Staals

Tukey Depth Histograms 15
Daniel Bertschinger, Jonas Passweg and Patrick Schnider

Enclosing Depth and other Depth Measures 16
Patrick Schnider

Geometric Dominating Sets - A Minimum Version of the No-Three-In-Line Problem 17
Oswin Aichholzer, David Eppstein and Eva-Maria Hainzl

Polyline Bundle Simplification on Trees 18
Yannick Bosch, Peter Schäfer and Sabine Storandt

Folding Polyiamonds into Octahedra 19
Eva Vanessa Bolle and Linda Kleist

Computing Optimal Virtual Camera Trajectories 20
Kerem Geva, Matthew Katz and Eli Packer

Polygon-Universal Graphs 21
Tim Ophelders, Ignaz Rutter, Bettina Speckmann and Kevin Verbeek

Rectilinear Steiner Trees in Narrow Strips 22
Henk Alkema and Mark de Berg

A density-based metric learning approach to geometric inference 23
Ximena Fernandez, Eugenio Borghini, Pablo Groisman and Gabriel Mindlin

Bicolored Path Embedding Problems in Protein Folding Models 24
Tianfeng Feng, Ryuhei Uehara and Giovanni Viglietta

On Voronoi diagrams of 1.5D terrains with multiple viewpoints 25
Vahideh Keikha and Maria Saumell

Upward Planar Drawings with Three Slopes 26
Jonathan Klawitter and Johannes Zink

Decomposing Polygons into Fat Components 27
Maike Buchin and Leonie Selbach

StreamTable: An Area Proportional Visualization for Tables with Flowing Streams 28
Jared Espenant and Debajyoti Mondal

An example of a randomized order-dependent time analysis in incremental construction 29
Evanthia Papadopoulou and Kolja Junginger

A Tail Estimate with Exponential Decay for the Randomized Incremental Construction of Search Structures 30
Joachim Gudmundsson and Martin P. Seybold

Shortest Paths in Portalgons 31
Maarten Löffler, Rodrigo Silveira and Frank Staals

Coordinating Programmable Matter via Shortest Path Trees 32
Irina Kostitsyna, Tom Peters and Bettina Speckmann

Long plane trees 33
Sergio Cabello, Michael Hoffmann, Katharina Klost, Wolfgang Mulzer and Josef Tkadlec

Terrain prickliness: theoretical grounds for low complexity viewsheds 34
Ankush Acharyya, Ramesh Jallu, Maarten Löffler, Gert Meijer, Maria Saumell, Rodrigo Silveira, Frank Staals and Hans Raj Tiwary

Tight Degree Bounds for Path-Greedy 5.19-Spanner on Convex Point Sets 35
William Evans and Lucca Siaudzionis

Crossing Numbers of Beyond-Planar Graphs Revisited 36
Nathan van Beusekom, Irene Parada and Bettina Speckmann

Reducing Moser's Square Packing Problem to a Bounded Number of Squares 37
Meike Neuwohner

On 4-Crossing-Families in Point Sets and an Asymptotic Upper Bound 38
Oswin Aichholzer, Jan Kyncl, Manfred Scheucher and Birgit Vogtenhuber

Collapses of higher order Delaunay complexes 39
Mickaël Buchet, Bianca B. Dornelas and Michael Kerber

Nearest-Neighbor Decompositions of Drawings 40
Jonas Cleve, Nicolas Grelier, Kristin Knorr, Maarten Löffler, Wolfgang Mulzer and Daniel Perz

Hardness of Recognition and 3-Coloring of L-graphs 41
Petr Chmel and Vít Jelínek

Coloring Circle Arrangements: New 4-Chromatic Planar Graphs 42
Man Kwun Chiu, Stefan Felsner, Manfred Scheucher, Felix Schröder, Raphael Steiner and Birgit Vogtenhuber

Minimum-Error Triangulation is NP-hard 43
Anna Arutyunova, Anne Driemel, Jan-Henrik Haunert, Herman Haverkort, Petra Mutzel and Heiko Röglin

On the Queue-Number of Partial Orders 44
Stefan Felsner, Torsten Ueckerdt and Kaja Wille

Uncertainty-Region Lower Bounds for the Preprocessing Model of Uncertainty 45
Ivor van der Hoog, Irina Kostitsyna, Maarten Löffler and Bettina Speckmann

Approximating Independent Set and Dominating Set on VPG graphs 46
Esther Galby and Andrea Munaro

Covering a Curve with Subtrajectories 47
Hugo Akitaya, Frederik Brüning, Erin Chambers and Anne Driemel

Radial Level Planarity with Fixed Embedding 48
Guido Brückner and Ignaz Rutter

Row-based Rectangle Contact Representations of Triangular Grid Graphs 49
Martin Nöllenburg, Anaïs Villedieu and Jules Wulms

Recognition of Unit Disk Graphs for Caterpillars, Trees, and Outerplanar Graphs 50
Sujoy Bhore, Soeren Nickel and Martin Nöllenburg

On the Realizability of Free Space Diagrams 51
Maike Buchin, Leonie Ryvkin and Carola Wenk

Accelerating Amoebots via Reconfigurable Circuits 52
Michael Feldmann, Andreas Padalkin, Christian Scheideler and Shlomi Dolev

Coloring Drawings Of Graphs 53
Christoph Hertrich, Felix Schröder and Raphael Steiner

Local Complexity of Polygons 54
Fabian Klute, Meghana M. Reddy and Tillmann Miltzow

Minimum Link Fencing 55
Sujoy Bhore, Fabian Klute, Maarten Löffler, Soeren Nickel, Martin Nöllenburg and Anaïs Villedieu

Characterizing Universal Reconfigurability of Modular Pivoting Robots 56
Hugo Akitaya, Erik D. Demaine, Andrei Gonczi, Dylan Hendrickson, Adam Hesterberg, Matias Korman, Oliver Korten, Jayson Lynch, Irene Parada and Vera Sacristán

Rectangular Spiral Galaxies are Still Hard 57
Erik D. Demaine, Maarten Löffler and Christiane Schmidt

Deletion only Dynamic Connectivity for Disk Graphs 58
Haim Kaplan, Katharina Klost, Kristin Knorr, Wolfgang Mulzer and Liam Roditty

Maximum-Width Rainbow-Bisecting Empty Annulus 59
Sandip Banerjee, Arpita Baral and Priya Ranjan Sinha Mahapatra

Axis-Aligned Square Contact Representations 60
Andrew Nathenson

The Voronoi Diagram of Rotating Rays with applications to Floodlight Illumination 61
Carlos Alegria, Ioannis Mantas, Evanthia Papadopoulou, Marko Savić, Hendrik Schrezenmaier, Martin Suderland and Carlos Seara

The Fréchet Distance for Plane Graphs 62
Pan Fang and Carola Wenk

Sampling Hyperplanes and Revealing Disks 63
Haim Kaplan, Alexander Kauer, Wolfgang Mulzer and Liam Roditty

On Minimum-Complexity Graph Simplification 64
Omrit Filtser, Majid Mirzanezhad and Carola Wenk

On the Geometric Red-Blue Hitting Set Problem 65
Raghunath Reddy Madireddy and Supantha Pandit

Intersecting Disks using Two Congruent Disks 66
Byeonguk Kang, Jongmin Choi and Hee-Kap Ahn

Online Ply Maintenance 67
Vikrant Ashvinkumar, Patrick Eades, Maarten Löffler and Seeun William Umboh

Hypergraph Represention via Axis-Aligned Point-Subspace Cover 68
Oksana Firman and Joachim Spoerhase

Minimizing the Maximum Interference in Dual Power Sensor Networks 69
Aviad Baron

Outerstring graphs of girth at least five are3-colorable 70
Sandip Das, Joydeep Mukherjee and Uma Kant Sahoo

Disjoint Box Covering in a Rectilinear Polygon 71
Sujoy Bhore, Guangping Li, Martin Nöllenburg and Jules Wulms

Uncertain Curve Simplification 72
Kevin Buchin, Maarten Löffler, Aleksandr Popov and Marcel Roeloffzen

Keeping your Distance: Algorithms and Hardness (Invited Talk)

Anna Lubiw[1]

1 David R. Cheriton School of Computer Science, University of Waterloo

Abstract

Suppose each of us is given a region of the plane and must choose a position in that region, and our sad goal is to be as far from each other as possible. This is known as the *distant representatives problem,* and is related to packing problems. I will describe approximation algorithms and hardness results for the problem, including some new results on distant representatives when the regions are axis-aligned rectangles and line segments.

Biography

Anna Lubiw is a professor at the University of Waterloo in Canada, working in computational geometry and graph algorithms. She has co-chaired SoCG, Graph Drawing, and WADS, and is on the editorial boards of the Journal of Computational Geometry and the Journal of Graph Algorithms and Applications. She received her PhD in 1986 from the University of Toronto.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.

Crossing Lemmas for Multigraphs (Invited Talk)

János Pach[1]

1 Rényi Institute, Budapest and MIPT, Moscow

Abstract

According to the crossing lemma of Ajtai, Chvátal, Newborn, Szemerédi and Leighton (1981), the crossing number of any graph with n vertices and $m > 4n$ edges is at least $c \cdot \frac{m^3}{n^2}$. This result, which is tight up to the constant factor, has been successfully applied to a variety of problems in discrete and computational geometry, additive number theory, algebra, and elsewhere. In some applications, it is the bottleneck that one needs a lower bound on the crossing number of a multigraph rather than a graph. The aim of this talk is to review a number of recent attempts on how to deal with this challenge.

Biography

János Pach is Research Adviser at Rényi Institute, Budapest and Head of the Laboratory of Combinatorial and Geometric Structures at MIPT, Moscow. His main fields of interest are discrete and computational geometry, convexity, and combinatorics. He wrote more than 300 research papers. His books, "Research Problems in Discrete Geometry" (with Brass and Moser) and "Combinatorial Geometry" (with Agarwal) were translated into Japanese, Russian, and Chinese. He is co-editor-in-chief of Discrete & Computational Geometry and serves on the editorial boards of ten other professional journals. He was elected ACM Fellow (2011), member of Academia Europeae (2014), and AMS Fellow (2015). He was invited speaker at the International Congress of Mathematicians in Seoul (2014), and is Plenary Speaker at the European Congress of Mathematics in Portorož (2021).

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.

An universality theorem for stressable graphs in the plane (Invited Talk)

Gaiane Panina[1]

1 St. Petersburg Department of V. A. Steklov Institute of Mathematics RAS, St. Petersburg University

Abstract

Universality theorems (in the sense of N. Mnëv) claim that the realization space of a combinatorial object (a point configuration, a hyperplane arrangement, a convex polytope, etc.) can be arbitrarily complicated. We prove a universality theorem for a graph in the plane with a prescribed oriented matroid of stresses, that is the collection of signs of all possible equilibrium stresses of the graph. This research is motivated by the Grassmanian stratification (Gelfand, Goresky, MacPherson, Serganova) by thin Schubert cells, and by a recent series of papers on stratifications of configuration spaces of tensegrities (Doray, Karpenkov, Schepers, Servatius).

Biography

Gaiane Panina graduated from Leningrad State University in 1984, completed her PhD at St. Petersburg Department of V.A. Steklov Mathematical Institute. Her habilitation thesis (2007) is entitled "Virtual polytopes". Chronologically, her research interests are: convexity, polytopes, virtual polytopes, combinatorial rigidity, configuration spaces, Morse theory, discrete Morse theory, combinatorial models of moduli spaces, combinatorial geometry, universality. At present she runs special courses at the Mathematics and Computer Science faculty of St. Petersburg State University.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.

Canonical Forms = Persistence Diagrams (Tutorial)

Serguei Barannikov[1]

1 Skoltech, Paris Diderot University

Abstract

The tutorial is devoted to the following theorem: any filtered complex over a field $\mathbf{k}$ can be brought by a linear transformation preserving the filtration to canonical form, a canonically defined direct sum of indecomposable filtered complexes of two types: one-dimensional complexes with trivial differential $\partial(e_{t_i}) = 0$ and two-dimensional complexes with trivial homology $\partial(e_{s_j}) = e_{r_j}$. The proof of this theorem was first published in the speaker's 1994 paper "Framed Morse complex and its invariants", AMS, Advances in Soviet Mathematics, volume 21, pages 93–115 (1994). This classification theorem is usually referred to in applied mathematics as the Persistent Homology Main (or Structure, or Principal) Theorem. The proof is elementary and uses only the basics of a standard linear algebra engineering course. The algorithms of more than 10 different software platforms, that exist actually for computation of persistence diagrams, have at their cores the described in the mentioned 1994 paper algorithm bringing filtered complexes to the canonical form.

Biography

Serguei Barannikov received his PhD in Mathematics from the University of California, Berkeley (1999). After completion of his PhD Serguei Barannikov has worked for ten years at Ecole Normale Supérieure in Paris. Serguei Barannikov is a leading research scientist at Skolkovo Institute of Science and Technology, and also works as a researcher in mathematics at Paris Diderot University.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.

Using SAT Solvers in Combinatorics, Combinatorial Geometry, and Graph Drawing (Tutorial)

Manfred Scheucher[1]

1 Skoltech, Paris Diderot University

Abstract

In this tutorial, we discuss how modern SAT solvers can be used to tackle mathematical problems from various areas, in particular, Combinatorics, Combinatorial Geometry, and Graph Drawing. To give the audience a better understanding, which problems can be tackled in this fashion, we will discuss some problems where we succeeded with our SAT attacks.

The focus will not be on the discussed problems themselves, but on the techniques that we had to use to finally come to a solution and the underlying ideas. For example, while the naive SAT formulation might already lead to an answer for some questions, it might also be way to big for nowadays computers and one has to come up with an equivalent formulation or deal with necessary/sufficient conditions instead. Also further ideas might be required so that the SAT instances finally become solvable in reasonable time (e.g. additional constraints for statements which hold "without loss of generality"). In particular, for our SAT attack on universal point sets for planar graphs we had to combine four sophisticated tools which have proven to be powerful on their own in the past: complete enumeration of order types, complete enumeration of (planar) graphs, SAT solvers, and IP solvers.

We can certainly not adress all details in this tutorial, but we look forward to showing the audience how to adress (their) problems in Combinatorics, Combinatorial Geometry, and Graph Drawing via SAT models and solvers.

Literature:

1. K. Däubel, S. Jäger, T. Mütze, and M. Scheucher. On orthogonal symmetric chain decompositions. In Electronic Journal of Combinatorics 26(3), 2019. [arχiv:1810.09847]
2. T. Mütze and M. Scheucher. On L-shaped Point Set Embeddings of Trees: First Non-embeddable Examples. In Journal of Graph Algorithms and Applications 24(3), 2020. [arχiv:1807.11043]
3. M. Scheucher. On Disjoint Holes in Point Sets. In Computational Geometry: Theory and Applications 91, 2020. [arχiv:1807.10848]
4. M. Scheucher. A SAT attack on higher dimensional Erdős–Szekeres numbers. In preparation, 2021+.
5. M. Scheucher, H. Schrezenmaier, and R. Steiner. A Note On Universal Point Sets for Planar Graphs. In Journal of Graph Algorithms and Applications 24(3), 2020. [arχiv:1811.06482]

Biography

Manfred Scheucher did his Master's at Graz University of Technology, Austria, supervised by Prof. Oswin Aichholzer, and his PhD at Technische Universität Berlin, Germany, supervised with Prof. Stefan Felsner.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.

Many Order Types on Integer Grids of Polynomial Size*

Manfred Scheucher[1,2]

1 **Institut für Mathematik,**
Technische Universität Berlin, Germany,
{scheucher}@math.tu-berlin.de
2 **Fakultät für Mathematik und Informatik,**
FernUniversität in Hagen, Germany

Abstract

Two labeled point configurations $\{p_1, \ldots, p_n\}$ and $\{q_1, \ldots, q_n\}$ are of the same order type if, for every i, j, k, the triples (p_i, p_j, p_k) and (q_i, q_j, q_k) have the same orientation. In the 1980's, Goodman, Pollack and Sturmfels showed that (i) the number of order types on n points is of order $4^{n+o(n)}$, (ii) all order types can be realized with double-exponential integer coordinates, and that (iii) certain order types indeed require double-exponential integer coordinates. In 2018, Caraballo, Díaz-Báñez, Fabila-Monroy, Hidalgo-Toscano, Leaños, Montejano showed that at least $n^{3n+o(n)}$ order types can be realized on an integer grid of polynomial size. In this article, we improve their result by showing that at least $n^{4n+o(n)}$ order types can be realized on an integer grid of polynomial size, which is essentially best possible.

1 Introduction

A set of n labeled points $\{p_1, \ldots, p_n\}$ in the plane with $p_i = (x_i, y_i)$ induces a *chirotope*, that is, a mapping $\chi\colon [n]^3 \to \{+, 0, -\}$ which assigns an orientation $\chi(a, b, c)$ to each triple of points (p_a, p_b, p_c) with

$$\chi(a,b,c) = \operatorname{sgn} \det \begin{pmatrix} 1 & 1 & 1 \\ x_a & x_b & x_c \\ y_a & y_b & y_c \end{pmatrix}.$$

Geometrically this means $\chi(a, b, c)$ is positive (negative) if the point p_c lies to the left (right) of the directed line $\overrightarrow{p_a p_b}$ through p_a directed towards p_b. Figure 1 gives an illustration. We say that two point sets are *equivalent* if they induce the same chirotope and call the equivalence classes *order types*. An order type in which three or more points lie on a common line is called *degenerate*.

Goodman and Pollack [14] (cf. [19, Section 6.2]) showed the number of order types on n points is of order $\exp(4n \log n + O(n)) = n^{4n+o(n)}$. While the lower bound follows from a simple recursive construction of non-degenerate order types, the proof of the upper bound uses the Milnor–Thom theorem [20, 23] (cf. [22, 25]) - a powerful tool from real algebraic geometry. The precise number of non-degenerate order types has been determined for up to 11 points by Aichholzer, Aurenhammer, and Krasser [1, 2] (cf. [18]). For their investigations, they used computer-assistance to enumerate all "abstract" order types and heuristics to either find a point set representation or to decide non-realizability. Similar approaches have

* The author acknowledges support by the internal research funding "Post-Doc-Funding" from Technische Universität Berlin.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

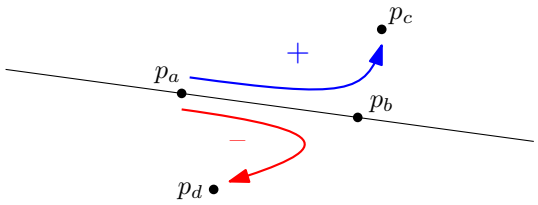


Figure 1 A chirotope with $\chi(a,b,c) = +$ and $\chi(a,b,d) = -$.

been taken by Fukuda, Miyata, and Moriyama [13] (cf. [12]) to investigate order types with degeneracies for up to 8 points. It is interesting to note that deciding realizability is an ETR-hard problem [21] and, since there are $\exp(\Theta(n^2))$ abstract order types (cf. [8] and [11]), most of them are non-realizable. For more details, we refer the interested reader to the handbook article by Felsner and Goodman [10].

Grünbaum and Perles [16, pp. 93–94] (cf. [4, pp. 355]) showed that there exist degenerate order types that are only realizable with irrational coordinates. Since we are mainly interested in order type representations with integer coordinates in this article, we will restrict our attention in the following to the non-degenerate setting.

Goodman, Pollack, and Sturmfels [15] showed that all non-degenerate order types can be realized with double-exponential integer coordinates and that certain order types indeed require double-exponential integer coordinates. Moreover, from their construction one can also conclude that $n^{4n+o(n)}$ order types on n points require integer coordinates of almost double-exponential size as outlined: For a slowly growing function $f : \mathbb{N} \to \mathbb{R}$ with $f(n) \to \infty$ as $n \to \infty$ and $m = \lfloor n/f(n) \rfloor \ll n$, we can combine each of the $(n-m)^{4(n-m)+o(n-m)} = n^{4n+o(n)}$ order types of $n-m$ points with the m-point construction from [15], which requires integer coordinates of size $\exp(\exp(\Omega(m)))$. Another infinite family that requires integer coordinates of super-polynomial size are the so-called *Horton sets* [3] (cf. [17]), which play a central role in the study of Erdős–Szekeres–type problems.

In 2018, Caraballo et al. [5] (cf. [6]) showed that at least $n^{3n+o(n)}$ non-degenerate order types can be realized on an integer grid of size $\Theta(n^{2.5}) \times \Theta(n^{2.5})$. In this article, we follow a similar approach as in [5] and improve their result by showing that $n^{4n+o(n)}$ order types can be realized on a grid of size $\Theta(n^4) \times \Theta(n^4)$.

▶ **Theorem 1.** *The number of non-degenerate order types which can be realized on an integer grid of size* $(3n^4) \times (3n^4)$ *is of order* $n^{4n+o(n)}$.

An important consequence of Theorem 1 is that a significant proportion of all n-point order types can be stored as point sets with $\Theta(\log n)$ bits per point. While the exponent in our $n^{4n+o(n)}$ bound is essentially best possible up to a lower-order error term, the question for the smallest constant c remains open for which $n^{4n+o(n)}$ order types can be realized on a grid of size $\Theta(n^c) \times \Theta(n^c)$.

Related Work

Besides the deterministic setting, also integer grid representations of "random" order types have been intensively studied in the last years. Fabila-Monroy and Huemer [9] and Devillers et al. [7] (cf. [24]) independently showed that, when a set $\{p_1, \ldots, p_n\}$ of real-valued points with $p_i = (x_i, y_i)$ are chosen uniformly and independently from the square $[0, n^{3+\epsilon}] \times [0, n^{3+\epsilon}]$, then the set $\{p'_1, \ldots, p'_n\}$ with rounded (integer-valued) coordinates $p'_i = ([x_i], [y_i])$ is of the same order type as the original set with high probability.

2 Proof of Theorem 1

Let n be a sufficiently large positive integer. It follows from Bertrand's postulate that we can find a prime number p satisfying $\frac{n}{2\lfloor \log n \rfloor} < p < \frac{n}{\lfloor \log n \rfloor}$. As an auxiliary point set, we let

$$Q_p = \{(x,y) \in \{1,\dots,p\}^2 \colon y = x^2 \mod p\}.$$

The point set Q_p contains p points from the $p \times p$ integer grid, and each two points have distinct x-coordinates. Moreover, Q_p is non-degenerate because, by the Vandermonde determinant, we have

$$\det \begin{pmatrix} 1 & 1 & 1 \\ a & b & c \\ a^2 & b^2 & c^2 \end{pmatrix} = (b-a)(c-a)(c-b) \neq 0 \mod p,$$

and hence $\chi(a,b,c) \neq 0$ for any pairwise distinct $a,b,c \in \{1,\dots,p\}$. In the following, we denote by $R(Q_p) = \{(y,x) \colon (x,y) \in Q_p\}$ the reflection of Q_p with respect to the line $x=y$.

Let $\alpha = 2n$ and $m = \alpha \cdot (2n^2 + n^3)$. For sufficiently large n, we have $2n^2 + n^3 \leq 2n^3$ and $m + 2p \leq 3n^4$. Our goal is to construct $4^{n+o(n)}$ different n-point order types on the integer grid

$$G = \{-p,\dots,m+p\} \times \{-p,\dots,m+p\}.$$

We start with placing four scaled and translated copies of Q_p, which we denote by D, U, L, R, as follows:

- To obtain D, we scale Q_p in x-direction by a factor $\alpha n \lfloor \log n \rfloor$ and translate by $(\alpha n^2, -p)$. All points from D have x-coordinates between αn^2 and $2\alpha n^2$ and y-coordinates between $-p$ and 0;
- To obtain U, we scale Q_p in x-direction by a factor $\alpha n \lfloor \log n \rfloor$ and translate by $(\alpha n^2, m)$. All points from U have x-coordinates between αn^2 and $2\alpha n^2$ and y-coordinates between m and $m+p$;
- To obtain L, we scale $R(Q_p)$ in y-direction by a factor $\alpha n \lfloor \log n \rfloor$ and translate by $(-p, \alpha n^2)$. All points from L have y-coordinates between αn^2 and $2\alpha n^2$ and x-coordinates between $-p$ and 0;
- To obtain R, we scale $R(Q_p)$ in y-direction by a factor $\alpha n \lfloor \log n \rfloor$ and translate by $(m, \alpha n^2)$. All points from R have y-coordinates between αn^2 and $2\alpha n^2$ and x-coordinates between m and $m+p$.

Each pair of points $(l,r) \in L \times R$ spans an *almost-horizontal* line-segment with absolute slope less than $\frac{1}{n}$. Similarly, each pair of points from $D \times U$ spans an *almost-vertical* line-segment with absolute reciprocal slope less than $\frac{1}{n}$. As depicted in Figure 2, these line-segments bound $(p^2-p)^2$ *almost-square* regions. Later, we will distribute the remaining $n-4p$ points among these almost-square regions in all possible way to obtain many different order types.

For every pair of distinct points d_1, d_2 from D, the x-distance between them is at least $\alpha n \lfloor \log n \rfloor$ and their y-distance is less than p. Hence, the absolute value of the slope of the line $\overline{d_1 d_2}$ is less than $\frac{p}{\alpha n \lfloor \log n \rfloor}$. Moreover, since d_1 and d_2 have non-positive y-coordinates and all points of G are at x-distance at most m from d_1 and d_2, the line $\overline{d_1 d_2}$ can only pass through points of G with y-coordinate less than $\frac{p \cdot m}{\alpha n \lfloor \log n \rfloor} \leq \alpha n^2$. (Recall that $m = \alpha \cdot (2n^2 + n^3) \leq 2\alpha n^3$ and $p \leq \frac{n}{\lfloor \log n \rfloor}$.) We conclude that every point from $U \cup L \cup R$ or from the almost-square regions lies strictly above the line $\overline{d_1 d_2}$. Similar arguments apply to lines spanned by pairs of points from U, L, and R, respectively. Note that, in particular, our construction has the property that for any point q from an almost-square region, the point set $D \cup U \cup L \cup R \cup \{q\}$ is non-degenerate.

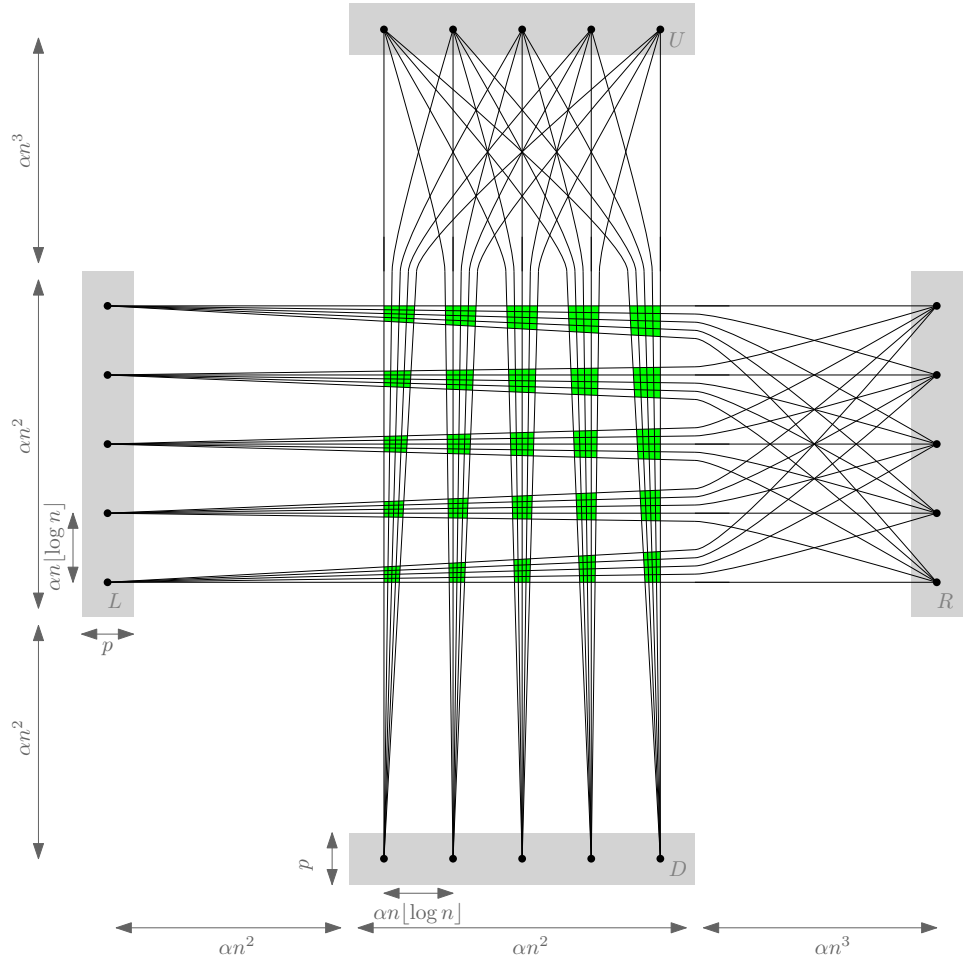


Figure 2 An illustration of the construction. The four copies D, U, L, R of Q_p are highlighted gray and the $(p^2 - p)^2$ almost-square regions are highlighted green.

Almost-square regions

Consider an almost-square region A with top-left vertex a, bottom-left vertex b, top-right vertex c, and bottom-right vertex d, as depicted in Figure 3. By our construction, the two almost-horizontal line-segments ℓ_1, ℓ_2 bounding A meet in a common end-point $l \in L$. Let $r_1, r_2 \in R$ denote the other end-points of ℓ_1 and ℓ_2, respectively, which have y-distance $\alpha n\lfloor\log n\rfloor$.

The point l has x-coordinate between $-p$ and 0, and the x-coordinates of the two points r_1 and r_2 are between m and $m + p$. Since we have chosen $m = \alpha \cdot (2n^2 + n^3)$ and assumed n to be sufficiently large, the x-distance between the points l and r_i (for $i = 1, 2$) is between αn^3 and $2\alpha n^3$. The point a lies on ℓ_1, b lies on ℓ_2, and the x-coordinates of both, a and b, are between αn^2 and $2\alpha n^2$. Hence, we can bound the y-distance δ between a and b by

$$\frac{1}{2}\alpha\lfloor\log n\rfloor = \alpha n\lfloor\log n\rfloor \cdot \frac{\alpha n^2}{\alpha 2n^3} \leq \delta \leq \alpha n\lfloor\log n\rfloor \cdot \frac{\alpha 2n^2}{\alpha n^3} = 2\alpha\lfloor\log n\rfloor.$$

Moreover, since a and b lie on an almost-vertical line (i.e., absolute reciprocal slope less than $\frac{1}{n}$), the x-distance between a and b is less than $\frac{2\alpha\lfloor\log n\rfloor}{n}$. An analogous argument applies to the pairs (a, c), (b, d), and (c, d), and hence we can conclude that the almost-square region A

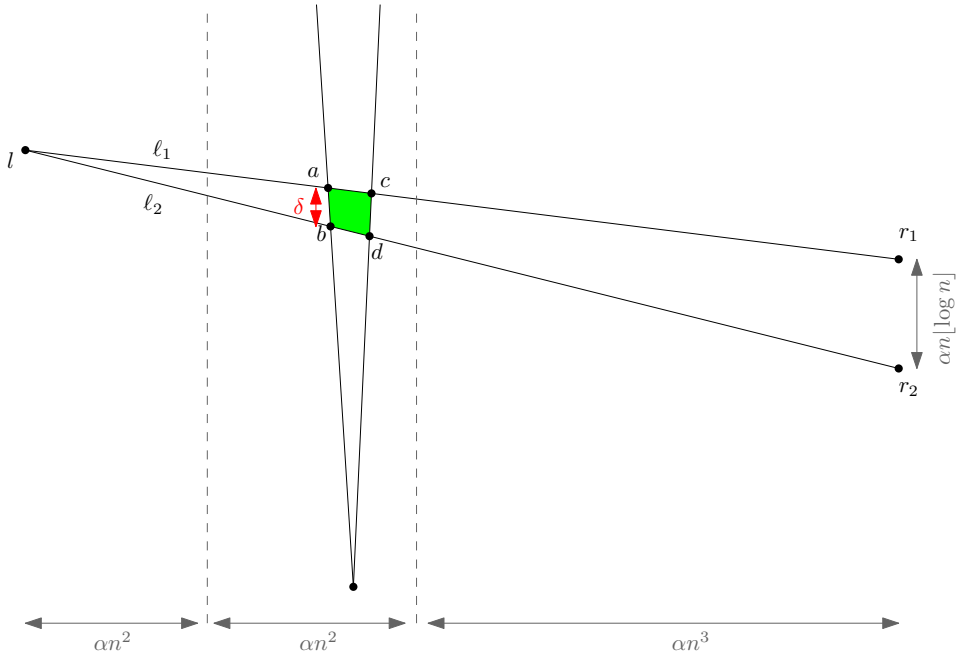


Figure 3 An almost-square region.

contains at least

$$\left(\frac{1}{2}\alpha\lfloor\log n\rfloor - \frac{4\alpha\lfloor\log n\rfloor}{n}\right)^2 \geq \frac{1}{5}\left(\alpha\lfloor\log n\rfloor\right)^2$$

points from the integer grid G, provided that n is sufficiently large.

Placing the remaining points

We have already placed p points in each of the four sets D, U, L, R. For each of the remaining $n-4p$ points, we can iteratively choose one of the $(p^2-p)^2$ almost-square regions and place it, unless our point set becomes degenerate. To deal with these degeneracy-issue, we denote an almost-square region A *alive* if there is at least one point from A which we can add to our current point configuration while preserving non-degeneracy. Otherwise we call A *dead*.

Having k points placed ($4p \leq k \leq n-1$), these k points determine $\binom{k}{2}$ lines which might *kill* points from our integer grid and some almost-square regions become *dead*. That is, if we add another point that lies on one of these $\binom{k}{2}$ lines to our point configuration, we clearly have a degenerate order type.

To obtain a lower bound on the number of alive almost-square regions, note that all almost-square regions lie in an $(\alpha n^2) \times (\alpha n^2)$ square and that each of the $\binom{k}{2}$ lines kills at most αn^2 grid points from almost-square regions. Moreover, since each almost-square region contains at least $\frac{1}{5}\left(\alpha\lfloor\log n\rfloor\right)^2$ grid points, we conclude that the number of alive almost-square regions is at least

$$(p^2-p)^2 - \binom{n}{2} \cdot \frac{\alpha n^2}{\frac{1}{5}\left(\alpha\lfloor\log n\rfloor\right)^2} \geq \frac{1}{17}\left(\frac{n}{\lfloor\log n\rfloor}\right)^4$$

for sufficiently large n, since $\frac{n}{2\lfloor\log n\rfloor} \leq p \leq \frac{n}{\lfloor\log n\rfloor}$ and $\alpha = 2n$.

If we now place each of the remaining $n - 4p$ points in an almost-square region which is alive (one by one), we have at least

$$\left(\frac{1}{17}\left(\frac{n}{\lfloor \log n \rfloor}\right)^4\right)^{n-4p} = n^{4n-O\left(n\frac{\log\log n}{\log n}\right)}$$

possibilities for doing so. Each of these possibilities clearly gives us a different order type because, when we move a point q from one almost-square region into another, this point q moves over a line spanned by a pair $(l, r) \in L \times R$ or $(d, u) \in D \times U$, and this affects $\chi(l, r, q)$ or $\chi(d, u, q)$, respectively. This completes the proof of Theorem 1.

References

1 O. Aichholzer, F. Aurenhammer, and H. Krasser. Enumerating order types for small point sets with applications. *Order*, 19(3):265–281, 2002.

2 O. Aichholzer and H. Krasser. Abstract order type extension and new results on the rectilinear crossing number. *Computational Geometry: Theory and Applications*, 36(1):2–15, 2006.

3 L. Barba, F. Duque, R. Fabila-Monroy, and C. Hidalgo-Toscano. Drawing the Horton set in an integer grid of minimum size. *Computational Geometry*, 63:10–19, 2017.

4 A. Björner, M. Las Vergnas, N. White, B. Sturmfels, and G. M. Ziegler. *Oriented Matroids*, volume 46 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 2 edition, 1999.

5 L. E. Caraballo, J.-M. Díaz-Báñez, R. Fabila-Monroy, C. Hidalgo-Toscano, J. Leaños, and A. Montejano. On the number of order types in integer grids of small size, 2018. arXiv:1811.02455.

6 L. E. Caraballo, J.-M. Díaz-Báñez, R. Fabila-Monroy, C. Hidalgo-Toscano, J. Leaños, and A. Montejano. On the number of order types in integer grids of small size. *Computational Geometry*, 95:101730, 2021.

7 O. Devillers, P. Duchon, M. Glisse, and X. Goaoc. On order types of random point sets, 2018. arXiv:1812.08525.

8 A. Dumitrescu and R. Mandal. New lower bounds for the number of pseudoline arrangements. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'19, pages 410–425. SIAM, 2019.

9 R. Fabila-Monroy and C. Huemer. *Order types of random point sets can be realized with small integer coordinates*, pages 73–76. 2017.

10 S. Felsner and J. E. Goodman. Pseudoline Arrangements. In Toth, O'Rourke, and Goodman, editors, *Handbook of Discrete and Computational Geometry*. CRC Press, third edition, 2018.

11 S. Felsner and P. Valtr. Coding and counting arrangements of pseudolines. *Discrete & Computational Geometry*, 46(3), 2011.

12 L. Finschi and K. Fukuda. Generation of oriented matroids–a graph theoretical approach. *Discrete & Computational Geometry*, 27(1):117–136, 2002.

13 K. Fukuda, H. Miyata, and S. Moriyama. Complete enumeration of small realizable oriented matroids. *Discrete & Computational Geometry*, 49(2):359–381, 2013.

14 J. E. Goodman and R. Pollack. Multidimensional sorting. *SIAM Journal on Computing*, 12(3):484–507, 1983.

15 J. E. Goodman, R. Pollack, and B. Sturmfels. Coordinate representation of order types requires exponential storage. In *Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing (STOC '89)*, pages 405–410. Association for Computing Machinery, 1989.

16 B. Grünbaum. *Convex Polytopes*. Springer, 2003.

17 J. Horton. Sets with no empty convex 7-gons. *Canadian Mathematical Bulletin*, 26:482–484, 1983.

18 D. E. Knuth. *Axioms and Hulls*, volume 606 of *LNCS*. Springer, 1992.

19 J. Matoušek. *Lectures on Discrete Geometry*. Springer, 2002.

20 J. Milnor. On the Betti numbers of real varieties. *Proceedings of the AMS*, 15:275–280, 1964.

21 N. E. Mnëv. The universality theorems on the classification problem of configuration varieties and convex polytopes varieties. In O. Y. Viro and A. M. Vershik, editors, *Topology and Geometry – Rohlin Seminar*, volume 1346 of *LNM*, pages 527–543. Springer, 1988.

22 I. G. Petrovskiĭ and O. A. Oleĭnik. On the topology of real algebraic surfaces. *Izvestiya Akad. Nauk SSSR. Ser. Mat.*, 13:389–402, 1949. In Russian.

23 R. Thom. *Sur L'Homologie des Variétés Algébriques Réelles*, pages 255–265. Princeton University Press, 1965.

24 I. van der Hoog, T. Miltzow, and M. van Schaik. Smoothed analysis of order types, 2019. arXiv:1907.04645.

25 H. Warren. Lower bounds for approximation by nonlinear manifolds. *Transactions of the AMS*, 133:167–178, 1968.

Tight bounds on the expected number of holes in random point sets*

Martin Balko[1], Manfred Scheucher[2,3], and Pavel Valtr[1]

1 **Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University, Czech Republic**
{balko}@kam.mff.cuni.cz
2 **Institut für Mathematik, Technische Universität Berlin, Germany**
{scheucher}@math.tu-berlin.de
3 **Fakultät für Mathematik und Informatik, FernUniversität in Hagen, Germany**

Abstract

For integers $d \geq 2$ and $k \geq d+1$, a *k-hole* in a set S of points in general position in $\mathbb{R}^d$ is a k-tuple of points from S in convex position such that the interior of their convex hull does not contain any point from S. For a convex body $K \subseteq \mathbb{R}^d$ of unit d-dimensional volume, we study the expected number $EH^K_{d,k}(n)$ of k-holes in a set of n points drawn uniformly and independently at random from K.

We prove an asymptotically tight lower bound on $EH^K_{d,k}(n)$ by showing that, for all fixed integers $d \geq 2$ and $k \geq d+1$, the number $EH^K_{d,k}(n)$ is at least $\Omega(n^d)$. For some small holes, we even determine the leading constant $\lim_{n\to\infty} n^{-d} EH^K_{d,k}(n)$ exactly. We improve the currently best known lower bound on $\lim_{n\to\infty} n^{-d} EH^K_{d,d+1}(n)$ by Reitzner and Temesvari (2019) and we show that our new bound is tight for $d \leq 3$. In the plane, we show that the constant $\lim_{n\to\infty} n^{-2} EH^K_{2,k}(n)$ is independent of K for every fixed $k \geq 3$ and we compute it exactly for $k = 4$, improving earlier estimates by Fabila-Monroy, Huemer, and Mitsche (2015) and by the authors (2020).

1 Introduction

For a positive integer d, let S be a set of points from $\mathbb{R}^d$ in *general position*. That is, no $d+1$ points from S lie on a k-dimensional affine subspace of $\mathbb{R}^d$. Throughout the paper we only consider point sets that are finite and in general position.

A point set P is in *convex position* if no point from P is contained in the convex hull of the remaining points from P. A *k-hole* H in S is a set of k points from S in convex position such that the convex hull $\mathrm{conv}(H)$ of H does not contain any point of S in its interior.

The study of k-holes in point sets was initiated by Erdős [7], who asked whether, for each $k \in \mathbb{N}$, every sufficiently large point set in the plane contains a k-hole. This was known to be

* M. Balko was supported by the grant no. 18-19158S of the Czech Science Foundation (GAČR), by the Center for Foundations of Modern Computer Science (Charles University project UNCE/SCI/004), and by the PRIMUS/17/SCI/3 project of Charles University. This article is part of a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 810115). M. Scheucher was partially supported by DFG Grant FE 340/12-1 and by the internal research funding "Post-Doc-Funding" from Technische Universität Berlin. We also gratefully acknowledge support from the internal research program IFFP 2016–2020 by the FernUniversiät in Hagen. P. Valtr was supported by the grant no. 18-19158S of the Czech Science Foundation (GAČR) and by the PRIMUS/17/SCI/3 project of Charles University. We thank Sophia and Jonathan Rau for helping with the computation in the proof of Theorem 2.4.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

true for $k \leq 5$, but, in the 1980s, Horton [11] constructed arbitrarily large point sets without 7-holes. The question about the existence of 6-holes was a longstanding open problem until 2007, when Gerken [10] and Nicolas [15] showed that every sufficiently large set of points in the plane contains a 6-hole.

The existence of k-holes was considered also in higher dimensions. Valtr [20] showed that, for $k \leq 2d+1$, every sufficiently large set of points in $\mathbb{R}^d$ contains a k-hole. He also constructed arbitrarily large sets of points in $\mathbb{R}^d$ that do not contain any k-hole with $k > 2^{d-1}(P(d-1)+1)$, where $P(d-1)$ denotes the product of the first $d-1$ prime numbers. Very recently Bukh, Chao, and Holzman [6] improved this construction.

Estimating the number of k-holes in point sets in $\mathbb{R}^d$ attracted a lot of attention; see [1]. In particular, it is well-known that the minimum number of $(d+1)$-holes (also called *empty simplices*) in sets of n points in $\mathbb{R}^d$ is of order $O(n^d)$. This is tight, as every set of n points in $\mathbb{R}^d$ contains at least $\binom{n-1}{d}$ $(d+1)$-holes [3, 12].

The tight upper bound $O(n^d)$ can be obtained by considering random point sets drawn from a convex body. More formally, a *convex body* in $\mathbb{R}^d$ is a compact convex subset of $\mathbb{R}^d$ with a nonempty interior. We use λ_d to denote the d-dimensional Lebesgue measure on $\mathbb{R}^d$ and $\mathcal{K}_d$ to denote the set of all convex bodies in $\mathbb{R}^d$ of volume $\lambda_d(K) = 1$. For an integer $k \geq d+1$ and a convex body $K \in \mathcal{K}_d$, let $EH^K_{d,k}(n)$ be the expected number of k-holes in a set S of n points chosen uniformly and independently at random from K. Note that S is in general position with probability 1.

Bárány and Füredi [3] proved the upper bound $EH^K_{d,d+1}(n) \leq (2d)^{2d^2} \cdot \binom{n}{d}$ for every $K \in \mathcal{K}_d$. Valtr [21] improved this bound in the plane by showing $EH^K_{2,3}(n) \leq 4\binom{n}{2}$ for any $K \in \mathcal{K}_2$. Very recently, Reitzner and Temesvari [16, Theorem 1.4] showed that this bound on $EH^K_{2,3}(n)$ is asymptotically tight for every $K \in \mathcal{K}_2$. This follows from their more general bounds $\lim_{n\to\infty} n^{-2} EH^K_{2,3}(n) = 2$ and

$$\frac{2}{d!} \leq \lim_{n\to\infty} n^{-d} EH^K_{d,d+1}(n) \leq \frac{d}{(d+1)} \frac{\kappa_{d-1}^{d+1} \kappa_{d^2}}{\kappa_d^{d-1} \kappa_{(d-1)(d+1)}} \tag{1}$$

for $d \geq 2$, where $\kappa_d = \pi^{\frac{d}{2}} \Gamma(\frac{d}{2}+1)^{-1}$ is the volume of the d-dimensional Euclidean unit ball. Moreover, the upper bound in (1) holds with equality in the case $d = 2$, and if K is a d-dimensional ellipsoid with $d \geq 3$. Note that, by (1), there are absolute positive constants c_1, c_2 such that

$$d^{-c_1 d} \leq \lim_{n\to\infty} n^{-d} EH^K_{d,d+1}(n) \leq d^{-c_2 d}$$

for every $d \geq 2$ and $K \in \mathcal{K}_d$.

Considering general k-holes in random point sets in $\mathbb{R}^d$, the authors [2] recently proved that $EH^K_{d,k}(n) \leq O(n^d)$ for all fixed integers $d \geq 2$ and $k \geq d+1$ and every $K \in \mathcal{K}_d$. More precisely, we showed

$$EH^K_{d,k}(n) \leq 2^{d-1} \cdot \left(2d^{2d-1} \binom{k}{\lfloor d/2 \rfloor}\right)^{k-d-1} \cdot \frac{n(n-1)\cdots(n-k+2)}{(k-d-1)! \cdot (n-k+1)^{k-d-1}}. \tag{2}$$

In this paper, we also study the expected number $EH^K_{d,k}(n)$ of k-holes in random sets of n points in K. In particular, we derive a lower bound that asymptotically matches the upper bound (2) for all fixed values of k. Moreover, for some small holes, we even determine the leading constants $\lim_{n\to\infty} n^{-d} EH^K_{d,k}(n)$.

2 Our Results

Our main result is that for all fixed integers $d \geq 2$ and $k \geq d+1$ the number $EH^K_{d,k}(n)$ is in $\Omega(n^d)$, which matches the upper bound (2) by the authors [2] up to the leading constant.

▶ **Theorem 2.1.** *For all integers $d \geq 2$ and $k \geq d+1$, there are constants $C = C(d,k) > 0$ and $n_0 = n_0(d,k)$ such that, for every integer $n \geq n_0$ and every convex body $K \subseteq \mathbb{R}^d$ of unit volume, we have $EH^K_{d,k}(n) \geq C \cdot n^d$.*

In particular, we see that random point sets typically contain many k-holes no matter how large k is, as long as it is fixed. This contrasts with the fact that, for every $d \geq 2$, there is a number $t = t(d)$ and arbitrarily large sets of points in $\mathbb{R}^d$ without any t-holes [11, 20].

Theorem 2.1 together with (2) shows that $EH^K_{d,k}(n) = \Theta(n^d)$ for all fixed integers d and k and every $K \in \mathcal{K}^d$, which determines the asymptotic growth rate of $EH^K_{d,k}(n)$. We thus focus on determining the leading constants $\lim_{n\to\infty} n^{-d} EH^K_{d,k}(n)$.

For a convex body $K \subseteq \mathbb{R}^d$ (of a not necessarily unit volume), we use p^K_d to denote the probability that the convex hull of $d+2$ points chosen uniformly and independently at random from K is a d-simplex. That is, the probability that one of the $d+2$ points falls in the convex hull of the remaining $d+1$ points. The problem of computing p^K_d is known as the d-dimensional *Sylvester's convex hull problem* for K and it has been studied extensively. Let $p_d = \max_K p^K_d$, where the maximum is taken over all covnex bodies K $\subseteq \mathbb{R}^d$. We note that the maximum is achieved, since it is well-known that every affine-invariant continuous functional on the space of convex bodies attains a maximum.

First, we prove the following lower bound on the expected number $EH^K_{d,d+1}(n)$ of empty simplices in random sets of n points in K, which improves the lower bound from (1) by Reitzner and Temesvari [16] by a factor of d/p_{d-1}.

▶ **Theorem 2.2.** *For every integer $d \geq 2$ and every convex body $K \subseteq \mathbb{R}^d$ of unit volume, we have*

$$\lim_{n\to\infty} n^{-d} EH^K_{d,d+1}(n) \geq \frac{2}{(d-1)!p_{d-1}}.$$

Using the trivial fact $p_1 = 1$ with the inequality $EH^K_{2,3}(n) \leq 2(1+o(1))n^2$ proved by Valtr [21], we see that the leading constant in our estimate is asymptotically tight in the planar case. An old result of Blaschke [4, 5] implies that Theorem 2.2 is also asymptotically tight for simplices in $\mathbb{R}^3$.

▶ **Corollary 2.3.** *For every convex body $K \subseteq \mathbb{R}^3$ of unit volume, we have*

$$3 \leq \lim_{n\to\infty} n^{-3} EH^K_{3,4}(n) \leq \frac{12\pi^2}{35} \approx 3.38.$$

Moreover, the left inequality is tight if K is a tetrahedron and the right inequality is tight if K is an ellipsoid.

Note that, in contrast to the planar case, the leading constant in $EH^K_{3,4}(n)$ depends on the body K.

By Theorem 2.2, better upper bounds on p_{d-1} give stronger lower bounds on $EH^K_{d,d+1}(n)$. The problem of estimating p_d is equivalent to the problem of estimating the expected d-dimensional volume EV^K_d of the convex hull of $d+1$ points drawn from a convex body $K \subseteq \mathbb{R}^d$ uniformly and independently at random, since $p^K_d = \frac{(d+2)EV^K_d}{\lambda_d(K)}$ for every $K \in \mathcal{K}_d$; see [14, 18]. In the plane, Blaschke [4, 5] showed that EV^K_2 is maximized if K is a triangle,

which we use to derive the lower bound in Corollary 2.3. For $d \geq 3$, it is one of the major problems in convex geometry to decide whether EV_d^K is maximized if K is a simplex [19].

Besides empty simplices, we also consider larger k-holes. The expected number $EH_{2,4}^K(n)$ of 4-holes in random planar sets of n points was considered by Fabila-Monroy, Huemer, and Mitsche [9], who showed $EH_{2,4}^K(n) \leq 18\pi D^2 n^2 + o(n^2)$ for any $K \in \mathcal{K}_2$, where $D = D(K)$ is the diameter of K. Since we have $D \geq 2/\sqrt{\pi}$, by the Isodiametric inequality [8], the leading constant in their bound is at least 72 for any $K \in \mathcal{K}_2$. This result was strengthened by the authors [2] to $EH_{2,4}^K(n) \leq 12n^2 + o(n^2)$ for every $K \in \mathcal{K}_2$. Here we determine the leading constant in $EH_{2,4}^K(n)$ exactly.

▶ **Theorem 2.4.** *For every convex body $K \subseteq \mathbb{R}^2$ of unit area, we have*

$$\lim_{n\to\infty} n^{-2} EH_{2,4}^K(n) = 10 - \frac{2\pi^2}{3} \approx 3.420.$$

Our computer experiments support this result. We sampled random sets of n points from a square and from a disk and the average number of 4-holes was around $3.42n^2$ for $n = 25000$ in our experiments. The source code of our program is available on the supplemental website [17].

For larger k-holes in the plane, we do not determine the value $\lim_{n\to\infty} n^{-2} EH_{2,k}^K(n)$ exactly, but we can show that it does not depend on the convex body K. We recall that this is not true in larger dimensions already for empty simplices.

▶ **Theorem 2.5.** *For every integer $k \geq 3$, there is a constant $C = C(k)$ such that, for every convex body $K \subseteq \mathbb{R}^2$ of unit area, we have*

$$\lim_{n\to\infty} n^{-2} EH_{2,k}^K(n) = C.$$

The proof of our main result, Theorem 2.1, is quite technical. So is the proof of Theorem 2.2, which is based on the Blaschke–Petkantschin formula (see Theorem 7.2.7 in [19]) and the well-known Lebesgue's dominated convergence theorem. Therefore we decided to devote Section 3 to an illustration of the proofs of Theorems 2.4 and 2.5. We only sketch the idea of the proof for 3-holes, because the proof for k-holes becomes more technical as k grows, but the main underlying idea remains the same. The full proofs of our results can be found in the appendices.

2.1 Open problems

As we remarked earlier, any nontrivial upper bound on the probability p_{d-1} translates into a stronger lower bound on $\lim_{n\to\infty} n^{-d} EH_{d,d+1}^K(n)$. However, we are not aware of any such estimate on p_{d-1}. Kingman [13] showed

$$p_d^{B^d} = \frac{(d+2)\binom{d+1}{\frac{d+1}{2}}^{d+1}}{2^d \binom{(d+1)^2}{\frac{(d+1)^2}{2}}},$$

which is of order $d^{-\Theta(d)}$. We conjecture that the upper bound on p_d^K is of this order for any convex body from $\mathcal{K}_d$.

▶ **Conjecture 2.6.** *There is a constant $c > 0$ such that, for every $d \geq 2$, we have $p_d \leq d^{-cd}$.*

We also believe that our lower bound from Theorem 2.2 is tight for simplices in arbitrarily large dimension d, not only for $d \leq 3$.

▶ **Conjecture 2.7.** *For every $d \geq 2$, if K is a d-dimensional simplex of unit volume, then* $\lim_{n\to\infty} n^{-d} EH^K_{d,d+1}(n) = \frac{2}{(d-1)!p_{d-1}}$.

As remarked earlier, it is widely believed that p^K_d is maximized if K is a simplex. If this is true, then it follows from the proof of Theorem 2.2 that Conjecture 2.7 is true as well.

It might also be interesting to determine $\lim_{n\to\infty} n^{-2} EH^K_{2,k}(n)$ exactly for as many values $k > 4$ as possible. Recall that, by Theorem 2.5, the number $\lim_{n\to\infty} n^{-2} EH^K_{2,k}(n)$ is the same for all convex bodies $K \in \mathcal{K}_2$.

3 Sketch of the proof for empty simplices in planar point sets

We sketch the proof that the expected number of 3-holes in a set S of n points selected uniformly and independently at random from a convex body $K \subseteq \mathbb{R}^2$ of unit volume is $2n^2 + o(n^2)$. For two points p_i and p_j from S, we count the expected number of 3-holes in S where p_i and p_j determine the longest edge.

Without loss of generality we can assume that $p_i = (0, 0)$ and $p_j = (\ell, 0)$ for some $\ell > 0$, as otherwise we apply a suitable isometry to S. Let R be the set of points from $K \cap ([0, \ell] \times [-\frac{2}{\ell}, \frac{2}{\ell}])$ that are at distance at most ℓ from p_i and also from p_j. Note that the set R is convex. The third point p_k of the 3-hole satisfies $x(p_i) < x(p_k) < x(p_j)$, as otherwise p_ip_j is not the longest edge of the 3-hole. If $|y(p_k)| > \frac{2}{\ell}$, then the convex hull of the 3-hole has area larger than 1, which is impossible. Consequently, p_k lies in R. For a real number $y \in [-\frac{2}{\ell}, \frac{2}{\ell}]$, let I_y be the line segment formed by points $r \in R$ with $y(r) = y$. Note that $|I_y| \leq \ell$ for every y and that $|I_0| = \ell$; see Figure 1.

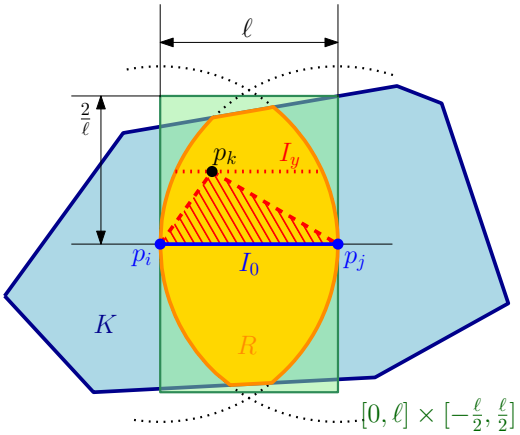


Figure 1 Sketch of the proof.

Since there are $n - 2$ candidates for p_k among $S \setminus \{p_i, p_j\}$, we can express the expected number of 3-holes in S where p_i and p_j determine the longest edge as

$$(n-2) \cdot \int_{-2/\ell}^{2/\ell} |I_y| \cdot Pr[p_ip_jp_k \text{ is empty in } S] \mathrm{d}y = (n-2) \cdot \int_{-2/\ell}^{2/\ell} |I_y| \cdot \left(1 - \frac{|y| \cdot \ell}{2}\right)^{n-3} \mathrm{d}y.$$

We now substitute $Y = yn$ and obtain

$$\frac{n-2}{n} \cdot \int_{-2n/\ell}^{2n/\ell} |I_{Y/n}| \cdot \left(1 - \frac{|Y| \cdot \ell}{2n}\right)^{n-3} \mathrm{d}Y.$$

By the Lebesgue dominated convergence theorem, we get for $n \to \infty$

$$2 \cdot \int_0^\infty |I_0| \cdot e^{-Y \cdot \ell/2} \mathrm{d}Y = 2 \cdot \int_0^\infty \ell \cdot e^{-Y \cdot \ell/2} \mathrm{d}Y = 4.$$

Since there are $\binom{n}{2}$ pairs $\{p_i, p_j\}$ in S, the expected number of 3-holes in S is $4(1+o(1)) \cdot \binom{n}{2} = 2n^2 + o(n^2)$ for n going to infinity.

References

1 O. Aichholzer, M. Balko, T. Hackl, J. Kynčl, I. Parada, M. Scheucher, P. Valtr, and B. Vogtenhuber. A superlinear lower bound on the number of 5-holes. *Journal of Combinatorial Theory. Series A*, 173:105236, 2020.

2 M. Balko, M. Scheucher, and P. Valtr. Holes and Islands in Random Point Sets. In *36th International Symposium on Computational Geometry (SoCG 2020)*, volume 164 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 14:1–14:16. Schloss Dagstuhl, 2020.

3 I. Bárány and Z. Füredi. Empty simplices in Euclidean space. *Canadian Mathematical Bulletin*, 30(4):436–445, 1987.

4 W. Blaschke. Lösung des 'Vierpunktproblems' von Sylvester aus der Theorie der geometrischen Wahrscheinlichkeiten. *Berichte über die Verhandlungen der Sächsischen Akademie der Wissenschaften zu Leipzig. Mathematisch-Physische Klasse*, 69:436–453, 1917.

5 W. Blaschke. *Vorlesungen über Differentialgeometrie und geometrische Grundlagen von Einsteins Relativitätstheorie II.* Die Grundlehren der mathematischen Wissenschaften. Springer, 1923.

6 B. Bukh, T. Chao, and R. Holzman. On convex holes in d-dimensional point sets. `http://arXiv.org/abs/2007.08972`, 2020.

7 P. Erdős. Some more problems on elementary geometry. *Australian Mathematical Society Gazette*, 5:52–54, 1978.

8 L. C. Evans and R. F. Gariepy. *Measure theory and fine properties of functions.* Textbooks in Mathematics. CRC Press, revised edition, 2015.

9 R. Fabila-Monroy, C. Huemer, and D. Mitsche. Empty non-convex and convex four-gons in random point sets. *Studia Scientiarum Mathematicarum Hungarica*, 52(1):52–64, 2015.

10 T. Gerken. Empty Convex Hexagons in Planar Point Sets. *Discrete & Computational Geometry*, 39(1):239–272, 2008.

11 J. Horton. Sets with no empty convex 7-gons. *Canadian Mathematical Bulletin*, 26:482–484, 1983.

12 M. Katchalski and A. Meir. On empty triangles determined by points in the plane. *Acta Mathematica Hungarica*, 51(3-4):323–328, 1988.

13 J. F. C. Kingman. Random secants of a convex body. *J. Appl. Probability*, 6:660–672, 1969.

14 V. Klee. Research Problems: What is the Expected Volume of a Simplex Whose Vertices are Chosen at Random from a Given Convex Body? *American Mathematical Monthly*, 76(3):286–288, 1969.

15 M. C. Nicolas. The Empty Hexagon Theorem. *Discrete & Computational Geometry*, 38(2):389–397, 2007.

16 M. Reitzner and D. Temesvari. Stars of empty simplices. `http://arxiv.org/abs/1808.08734`, 2019.

17 M. Scheucher. Supplemental program for computer experiments. `http://page.math.tu-berlin.de/~scheuch/suppl/holes_in_random_sets/test_planar_holes.py`.

18 R. Schneider. Random approximation of convex sets. *Journal of Microscopy*, 151(3):211–227, 1988.

19 R. Schneider and W. Weil. *Stochastic and integral geometry*. Probability and its Applications. Springer, 2008.

20 P. Valtr. Sets in $\mathbb{R}^d$ with no large empty convex subsets. *Discrete Mathematics*, 108(1):115–124, 1992.

21 P. Valtr. On the minimum number of empty polygons in planar point sets. *Studia Scientiarum Mathematicarum Hungarica*, pages 155–163, 1995.

Obstructing Classification via Projection

Pantea Haghighatkhah[1], Wouter Meulemans[1], Bettina Speckmann[1], Jérôme Urhausen[2], and Kevin Verbeek[1]

1 **TU Eindhoven, The Netherlands**
[p.haghighatkhah|w.meulemans|b.speckmann|k.a.b.verbeek]@tue.nl
2 **Utrecht University, The Netherlands**
j.e.urhausen@uu.nl

1 Introduction

Machine learning and data mining techniques are effective tools to classify large amounts of data. But they tend to preserve any inherent bias in the data, for example, with regards to gender or race. The identification and removal of bias receives significant attention and various approaches have been described in the machine learning literature using, for example, statistical methods [3], preprocessing [2, 4], or additional (possibly adversarial) models [6]. Abbasi *et al.* [1] recently introduced a geometric notion of stereotyping which postulates that bias is in some form encoded in the geometric or topological features of the learned model and that manipulating this geometry can remove the bias. In this paper we follow the same premise and study a possible geometric approach for bias removal.

We model the data as points in $\mathbb{R}^d$ which are labeled with binary-valued properties, and assume that it is "easy" to classify the data according to each property. Our goal is to obstruct the classification according to one property by a suitable projection to a lower-dimensional Euclidean space, while classification according to all other properties remains easy.

Formal problem statement Our input is a set of n points $P = \{p_1, \ldots, p_n\}$ in general position in $\mathbb{R}^d$. For convenience, we identify the points with their corresponding vector. For all points in P we are given k binary-valued *properties*, represented as functions $a_i \colon P \to \{-1, 1\}$ for $1 \leq i \leq k$. We denote the subset of points $p \in P$ with $a_i(p) = 1$ as P^i_+, and the subset of points $p \in P$ with $a_i(p) = -1$ as P^i_- for $1 \leq i \leq k$. For a point $p \in P$, we refer to the tuple $(a_1(p), \ldots, a_k(p))$ as the *label* of p. Note that there are 2^k different possible labels.

We assume that it is "easy" to classify the points in P according to the properties by using the point coordinates. Our goal is to compute a projection P' of P to lower dimensions such that the first property a_1 becomes hard to classify in P', and the other properties $a_2, \ldots, a_k$ remain easy to classify in P'. We denote by $P_- = P^1_-$ and $P_+ = P^1_+$ the point sets in which the special property a_1 is set to -1 and $+1$, respectively. Similarly, we use P'_- and P'_+ for the point sets P_- and P_+ after projection. Usually we consider a projection along a single unit vector w ($\|w\| = 1$), mapping points in $\mathbb{R}^d$ to points in $\mathbb{R}^{d-1}$. For $p_i \in P$, we denote its projection as $p'_i = p_i - (p_i \cdot w)w$ (here $(p_i \cdot w)$ is the dot product). Note that this projection simply restricts p'_i to a hyperplane in $\mathbb{R}^d$, rather than mapping it to $\mathbb{R}^{d-1}$. Sometimes we will consider projections along multiple vectors $w_1, \ldots, w_r$. In that case we assume that $\{w_j\}_{j=1}^r$ form an orthonormal system, such that we can write the projection as $p'_i = p_i - \sum_{j=1}^r (p_i \cdot w_j)w_i$. Again, we assume that p'_i still lies in $\mathbb{R}^d$, but is restricted to the $(d-r)$-dimensional flat that is orthogonal to $w_1, \ldots, w_r$ and passes through the origin.

We consider different models to define what is easy or hard to classify, based on some form of "separability" between two point sets. We say that a property a_i is separated in a point set P to indicate that P^i_- and P^i_+ are separated (see Figure 1 for an example in $\mathbb{R}^2$).

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

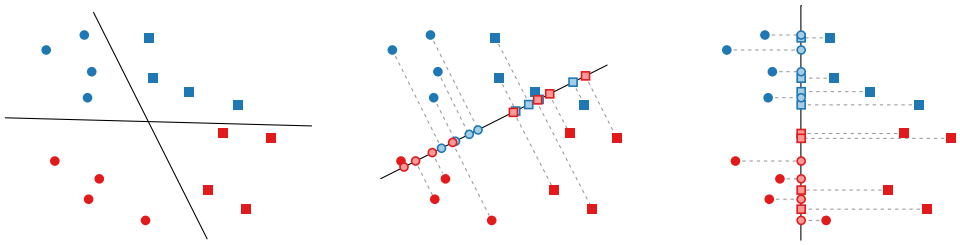


Figure 1 Left: data points with two linearly-separable properties: shape and color. Middle: a projection which keeps shape separated, but not color. Right: a projection with the opposite effect.

Contributions and organization In Section 2 we consider linear separability as the classification model. We first show that, if even one possible label is missing from P, then there may be no projection that eliminates the linear separability of a_1 whilst keeping the linear separability of the other properties. On the other hand, if all possible labels are present in the point set, then we show that it is always possible to achieve this goal. In Section 3 we introduce (b, c)-separability, which is a generalization of linear separability. Although a single projection is no longer sufficient to avoid (b, c)-separability of a_1 after projection, we show that the number of projections needed to achieve this is linked to the Helly number of the respective separability predicate. We then establish bounds on the Helly numbers of (b, c)-separability for specific values of b and c.

2 Linear separability

One of the machine learning techniques that use linear separability for classification are *support vector machines* (SVMs). SVMs compute the (optimal) hyperplane that separates two classes in the training data (if linearly separable), and use that hyperplane for further classifications. Linear separability is therefore a good first model to consider for classification.

For a point set P and property $a_i \colon P \to \{-1, 1\}$, we say that a_i is easy to classify on P if P^i_- and P^i_+ are (strictly) linearly separable; we say that a_i is hard to classify otherwise. Two point sets P and Q ($P, Q \subset \mathbb{R}^d$) are *linearly separable* if there exists a hyperplane H separating P from Q. The point sets are *strictly linearly separable* if we can additionally require that none of the points lie on H. Equivalently, the point sets P and Q are linearly separable if there exists a unit vector $v \in \mathbb{R}^d$ and constant $c \in \mathbb{R}$ such that $(v \cdot p) \leq c$ for all $p \in P$ and $(v \cdot q) \geq c$ for all $q \in Q$ (v is the normal vector of the hyperplane H). We say that P and Q are linearly separable *along* v. If the inequalities can be strict, then the point sets are strictly linearly separable.

Let $CH(P)$ denote the convex hull of a point set P. By definition, we have that $x \in CH(P)$ if and only if there exist coefficients $\lambda_i \geq 0$ such that $x = \sum_{i=1}^{n} \lambda_i p_i$ and $\sum_{i=1}^{n} \lambda_i = 1$. The following lemma is illustrated in Figure 2.

▶ **Lemma 1.** *Let P and Q be two point sets. If we project both P and Q along a unit vector w to obtain P' and Q', then P' and Q' are not strictly linearly separable iff there exists a line ℓ parallel to w that intersects both $CH(P)$ and $CH(Q)$. If ℓ intersects the interior of $CH(P)$ or $CH(Q)$, then P' and Q' are not linearly separable.*

Assume now that the properties $a_1, \ldots, a_k$ are strictly linearly separable in P. Can we project P along a unit vector w so that $a_2, \ldots, a_k$ are still strictly linearly separable in P', but a_1 is not? We consider two variants: (1) *separation preserving* and (2) *separability*

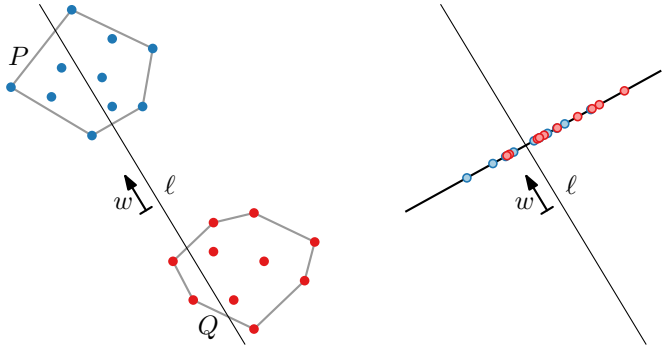


Figure 2 Line ℓ intersects both $CH(P)$ and $CH(Q)$; after projection the convex hulls intersect.

preserving projections. The former preserves a fixed set of separating hyperplanes $H_2, \ldots, H_k$ for properties $a_2, \ldots, a_k$, the latter preserves only separability of $a_2, \ldots, a_k$.

Lemma 2 proves that there exist point sets using only $2^k - 1$ possible labels for which every separability preserving projection also keeps a_1 strictly linearly separable after projection. The idea is to use the properties $a_2, \ldots, a_k$ to sufficiently restrict the direction of a separability preserving projection to make it impossible for this projection to eliminate the linear separability of a_1. A simple example for $d = k = 2$ is shown in Figure 3.

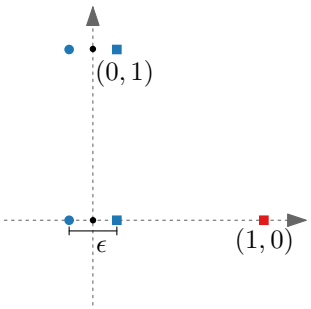


Figure 3 To keep a_2 (shape) linearly separable after projection, the projection vector w should be nearly vertical, but then a_1 (color) will also remain linearly separable.

▶ **Lemma 2.** *For all $k > 1$ and $d \geq k$, there exist point sets P in $\mathbb{R}^d$ with properties $a_1, \ldots, a_k$ using $2^k - 1$ labels such that any separability preserving projection along a unit vector w also keeps a_1 strictly linearly separable after projection.*

We now assume that all 2^k labels are used in P. Note that this assumption directly implies that $d \geq k$: take any set of k separating hyperplanes $H_1, \ldots, H_k$ for the k properties and consider the arrangement formed by the hyperplanes in $\mathbb{R}^d$. Clearly, all points in the same cell of the arrangement must have the same label. However, it is well-known that it is not possible to create 2^k cells in $\mathbb{R}^d$ with only k hyperplanes if $d < k$. This has also interesting implications for the case when $d = k$: if we apply a separation preserving projection to P, then a_1 cannot be linearly separable in P', since P' is embedded in $\mathbb{R}^{k-1}$.

We show that, if $d \geq k$, then there always exists a separation preserving projection that eliminates the strict linear separability of a_1. Our proof uses a natural variant of both Carathéodory's and Radon's theorem; see also Figure 4.

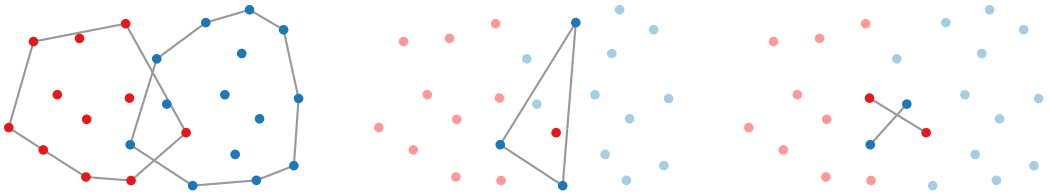


Figure 4 Lemma 3 in 2D: only 4 points are needed to construct two intersecting convex hulls.

▶ **Lemma 3.** *Let P and Q be two points sets in $\mathbb{R}^d$ such that $CH(P) \cap CH(Q) \neq \emptyset$. Then there exist subsets $P^* \subseteq P$ and $Q^* \subseteq Q$ such that $CH(P^*) \cap CH(Q^*) \neq \emptyset$ and $|P^*| + |Q^*| = d + 2$.*

▶ **Theorem 4.** *If P is a point set in $\mathbb{R}^d$ with k $(d \geq k)$ properties $a_1, \ldots, a_k$ using all 2^k labels, then there exists a separation preserving projection along a unit vector w that eliminates the strict linear separability of a_1.*

Proof. We provide an explicit construction of the vector w. Let $H_2, \ldots, H_k$ be any separating hyperplanes for each of the properties $a_2, \ldots, a_k$ in P, respectively. Let v_i be the normal of hyperplane H_i for $2 \leq i \leq k$, and let $A \subset \mathbb{R}^d$ be the $(k-1)$-dimensional linear subspace spanned by $v_2, \ldots, v_k$. Furthermore, let $H^* = \bigcap_{i=2}^{k} H_i$ be the $(d-k+1)$-dimensional flat that is the intersection of the separating hyperplanes. Note that a projection along a vector w is separation preserving if and only if w is parallel to H^*. Let $T(p)$ be the point obtained by performing an orthogonal projection of a point $p \in P$ onto A. For ease of argument, we also directly apply an affine transformation that maps H^* (which intersects A in one point) to the origin, and maps $v_2, \ldots, v_k$ to the standard basis vectors of $\mathbb{R}^{k-1}$.

Now define $Q_- = \{T(p) \mid p \in P_-\}$ and $Q_+ = \{T(p) \mid p \in P_+\}$. By construction, since all labels are used by P, both Q_- and Q_+ must have a point in each orthant of $\mathbb{R}^{k-1}$. If a point set Q has a point in each orthant, then $CH(Q)$ must contain the origin; because if it does not, then there exists a vector v such that $(v \cdot q) > 0$ for all $q \in Q$. But there must exist a point $q^* \in Q$ whose sign for each coordinate is opposite from that of v (or zero), which means that $(v \cdot q^*) \leq 0$, a contradiction. Thus, both $CH(Q_-)$ and $CH(Q_+)$ contain the origin, and $CH(Q_-) \cap CH(Q_+) \neq \emptyset$. We now apply Lemma 3 to Q_- and Q_+ to obtain Q_-^* and Q_+^* consisting of $k+1$ points in total. Let $P^* \subseteq P$ be the corresponding set of original points that map to $Q_-^* \cup Q_+^*$. We can now construct w as follows. Pick a point $p^* \in P^*$, and let F_1 be the unique $(k-1)$-dimensional flat that contains the remaining points in P^*. Let F_2 be the flat obtained by translating H^* to contain p^*. Since F_1 is $(k-1)$-dimensional and F_2 is $(d-k+1)$-dimensional, $F_1 \cap F_2$ consists of a single point $r \in \mathbb{R}^d$. The desired projection vector is now simply $w = r - p^*$ (normalized if necessary).

We finally show that the constructed vector w has the correct properties. First of all, w is parallel to H^* by construction, and hence the projection along w is separation preserving. Second, since $r \in F_1$ and p^* is projected to coincide with r, all points in P^* will lie on the same $(k-1)$-dimensional flat F_1' after projection. Also, since w is orthogonal to A, there exists an affine map from $Q_-^* \cup Q_+^*$ to P^* (after projection). Thus, we obtain that $CH(P_-') \cap CH(P_+') \neq \emptyset$; in particular, the convex hulls must intersect on F_1'. This implies that a_1 is not strictly linearly separable after projection. ◀

3 Generalized separability

In this section we consider a generalization of linear separability for classification. One approach to achieve more complicated classification boundaries is to use clustering: the label

of a point is determined by the label of the "nearest" cluster. If we use more than one cluster per class, then the resulting classification is more expressive than classification by linear separation. This approach is also strongly related to nearest neighbor classification, another common machine learning technique. Our generalized definition of separability is inspired by such clustering-based classifications, with convex sets modeling the clusters.

Let P and Q be two point sets in $\mathbb{R}^d$. We say that P and Q are (b, c)-*separable* if there exist b convex sets $S_1, \ldots, S_b$ and c convex sets $T_1, \ldots, T_c$ such that for every point $p \in P$ we have that $p \in S = \bigcup_i S_i$, for every point $q \in Q$ we have that $q \in T = \bigcup_j T_j$, and that $S \cap T = \emptyset$ (see Figure 5). We can assume that $b \leq c$. Linear separability and $(1, 1)$-separability are equivalent.

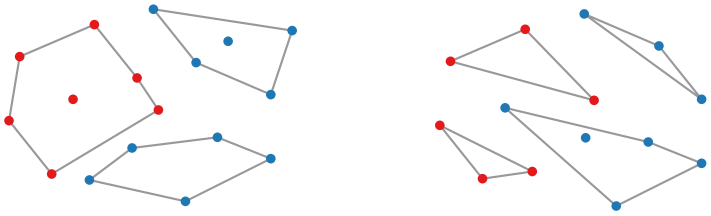


Figure 5 Left: two points sets P (red) and Q (blue) that are $(1, 2)$-separable, but not linearly separable. Right: two point sets that are $(2, 2)$-separable, but not $(1, x)$-separable for any value of x.

Given a point set P along with k properties $a_1, \ldots, a_k$, the goal is now to compute a separation preserving projection to a point set P' such that a_1 is not (b, c)-separable in P'. We again assume that all k properties are strictly linearly separable in P. To achieve this goal, we may need to project along multiple vectors $w_1, \ldots, w_r$.

We first generalize Theorem 4 to a generic separability predicate $F(P, Q)$ for point sets $P, Q \subset \mathbb{R}^d$.[1] A separability predicate is *well-behaved* if it satisfies the following conditions:

1. If $F(P, Q)$ does not hold, then $F(P', Q')$ does not hold, where P' and Q' are obtained by projecting P and Q along a single unit vector, respectively.
2. If $P' \subseteq P$ and $Q' \subseteq Q$, then $F(P, Q)$ implies $F(P', Q')$.
3. If $\mathcal{A}$ is an affine map, then $F(P, Q)$ holds if and only if $F(\mathcal{A}(P), \mathcal{A}(Q))$ holds.

Now assume that F has the *Helly-type property* [5]: if $F(P, Q)$ does not hold, then there exist small (bounded by a constant) subsets $P^* \subseteq P$ and $Q^* \subseteq Q$ such that $F(P^*, Q^*)$ also does not hold. The worst-case size of $|P^*| + |Q^*|$ often depends on the number of dimensions d of P and Q, and is referred to as the *Helly number* $m_F(d)$ of F. We can extend Theorem 4 to well-behaved separability predicates with the Helly-type property. The proof is analogous to the proof of Theorem 4, except that we now need to project $m_F(k-1)$ points to the same $(k-1)$-flat, and hence need $m_F(k-1) - k$ projections.

▶ **Theorem 5.** *Let P be a point set in $\mathbb{R}^d$ with k $(d \geq k)$ properties $a_1, \ldots, a_k$ and let F be a well-behaved separation predicate in $\mathbb{R}^d$. Either we can use at most $\min(m_F(k-1)-k, d-k+1)$ separation preserving projections to eliminate $F(P_-, P_+)$, or this cannot be achieved with any number of separation preserving projections.*

Given Theorem 5, we now focus on the Helly numbers of (b, c)-separability for specific b and c. Unfortunately, not every form of (b, c)-separability has the Helly-type property.

[1] We assume that F is defined independently from the dimensionality of P and Q (like (b, c)-separability). We do however require that P and Q are embedded in the same space.

▶ **Lemma 6.** *In $d \geq 2$ dimensions, $(1, 2)$-separability does not have the Helly-type property.*

Next, we consider $(1, \infty)$-separability. This means that one of the point sets, say P, must be covered with one convex set, but we can use arbitrarily many convex sets to cover Q. Equivalently, P and Q are $(1, \infty)$-separable if $CH(P) \cap Q = \emptyset$ or $P \cap CH(Q) = \emptyset$.

▶ **Lemma 7.** *In $d \geq 1$ dimensions, $(1, \infty)$-separability has Helly number $2d + 2$.*

4 Conclusion

We studied the use of projections for obstructing classification of high-dimensional Euclidean point data. Our results show that, if not all possible labels are present in the data, then it may not be possible to eliminate the linear separability of one property while preserving it for the other properties. This is not surprising if a property that we aim to keep is strongly correlated with the property we aim to hide. Nonetheless, one should be aware of this effect when employing projections for this purpose in practice. When going beyond linear separability, we see that the number of projections required to hide a property increases significantly in theory, and we expect a similar effect when using neural networks for classification in practice. In other words, projecting a dataset once (or few times) may not be sufficient to hide a property from a smart classifier. Projection, as a linear transformation, can however be effective in eliminating certain linear relations in the data.

References

1 Mohsen Abbasi, Sorelle A. Friedler, Carlos Scheidegger, and Suresh Venkatasubramanian. Fairness in representation: quantifying stereotyping as a representational harm. In *Proc. SIAM International Conference on Data Mining*, pages 801–809, 2019. doi:10.1137/1.9781611975673.90.

2 Alexander Amini, Ava P. Soleimany, Wilko Schwarting, Sangeeta N. Bhatia, and Daniela Rus. Uncovering and mitigating algorithmic bias through learned latent structure. In *Proc. AAAI/ACM Conference on AI, Ethics, and Society*, pages 289–295, 2019. doi:10.1145/3306618.3314243.

3 Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems*, pages 3315–3323, 2016.

4 Faisal Kamiran and Toon Calders. Data preprocessing techniques for classification without discrimination. *Knowledge and Information Systems*, 33:1–33, 2011. doi:10.1007/s10115-011-0463-8.

5 Rephael Wenger. Helly-type theorems and geometric transversals. In Jacob E. Goodman and Joseph O'Rourke, editors, *Handbook of Discrete and Computational Geometry, second edition*, pages 73–96. Chapman and Hall/CRC, 2004. doi:10.1201/9781420035315.ch4.

6 Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. Mitigating unwanted biases with adversarial learning. In *Proc. AAAI/ACM Conference on AI, Ethics, and Society*, pages 335–340, 2018. doi:10.1145/3278721.3278779.

Route Reconstruction from Traffic Flow via Representative Trajectories

Bram Custers[1], Wouter Meulemans[1], Bettina Speckmann[1], and Kevin Verbeek[1]

1 **TU Eindhoven, the Netherlands**
[b.a.custers|w.meulemans|b.speckmann|k.a.b.verbeek]@tue.nl

Related Version https://arxiv.org/abs/2012.05019

1 Introduction

Understanding human mobility patterns is an important aspect of traffic analysis and urban planning. Trajectory data provide detailed views on specific routes, but typically do not capture all traffic. On the other hand, loop detectors built into the road network capture all traffic at specific locations, but provide no information on the individual routes. Given a set of loop-detector data as well as a (small) set of representative trajectories, we investigate how one can effectively combine these two to create a more complete picture of the underlying mobility patterns. Specifically, we want to reconstruct a realistic set of routes from the loop-detector data, using the given trajectories as representatives of typical behavior.

We model the loop-detector data as a network flow field that needs to be covered by the reconstructed routes and we capture the realism of the routes via the strong Fréchet distance to the representative trajectories. The full version discussed our modeling decisions in detail; we summarize the resulting definitions, notation, and formal problem statement in Section 2. Several forms of the algorithmic problem are NP-hard. Hence we explore heuristic approaches which decompose the flow well, while following the representative trajectories to varying degrees. In Section 3, we propose an iterative *Fréchet Routes* (FR) heuristic which generates candidate routes which have bounded Fréchet distance to the representative trajectories. We also describe a variant of multi-commodity min-cost flow (MCMCF) which is only loosely coupled to the trajectories.

In Section 4 we experimentally evaluate our approaches in comparison to a global min-cost flow (GMCF), which is essentially agnostic to the representative trajectories. To make meaningful claims in terms of quality, we derive a ground truth by map-matching real-world trajectories and by generating synthetic routes in a real-world network. We find that GMCF explains the flow best, but produces a large number of often nonsensical routes (significantly more than the ground truth). MCMCF produces a large number of mostly realistic routes which explain the flow reasonably well. In contrast, FR produces much smaller sets of realistic routes which still explain the flow well, at the cost of a higher running time.

Related work Our problem is closely related to flow decomposition [1]. Given a set of paths that decompose the flow, it is NP-hard to determine the correct integral coefficients for each path [10]. Minimizing the number of paths in a decomposition is also NP-hard [14], and thus various approximation algorithms have been developed [6].

Reconstructing a route (in a network) given a GPS trajectory is referred to as map-matching [2, 7, 8, 12, 15], see also the survey by Quddus *et al.* [13]. Of specific relevance to our work is the result by Alt *et al.* [2] which solves map-matching under the Fréchet distance (see Section 3). For simple routes, map-matching is NP-hard for various distance measures [4, 11], though on a grid routes with bounded distance can be found efficiently [4].

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

2 Preliminaries

Our input has three components: (1) a road network, given as a graph $\mathcal{G} = (V, E)$ with n vertices and m edges, embedded in $\mathbb{R}^2$; (2) a set of representative trajectories $\mathcal{T}$, each encoded by a sequence $\langle p_1, \ldots, p_n \rangle$ of measurements, with $p_i = (x_i, y_i) \in \mathbb{R}^2$; and (3) loop-detector data. We model the loop-detector data as a *flow field*: a function $\phi : E \to \mathbb{R}_{\geq 0}$ that assigns a traffic volume to each edge in the road network $\mathcal{G}$.

Our goal is to reconstruct a realistic set of routes from the loop-detector data, using the trajectories as representatives of typical behavior. A *route* $P = \langle e_1, \ldots, e_k \rangle$ is a sequence of edges that encode the traversed path in the road network $\mathcal{G}$. We say that a route is *simple* if it visits every vertex in $\mathcal{G}$ at most once. For ease of notation, we introduce the function $M(P, e)$ that indicates how often the edge $e \in E$ is traversed in the route P.

We define a *reconstruction* $\overline{\mathcal{P}}$ of the flow field ϕ as a pair $\overline{\mathcal{P}} = (\mathcal{P}, c)$ with a base set of routes $\mathcal{P}$ (the *basis*) and fractional *coefficients* $c \colon \mathcal{P} \to \mathbb{R}_{\geq 0}$. A reconstruction $(\mathcal{P}, c)$ defines a *flow* $(f_{(\mathcal{P},c)}, S_{\mathcal{P}}, T_{\mathcal{P}})$ as $f_{(\mathcal{P},c)}(e) = \sum_{P \in \mathcal{P}} M(P, e)c(P)$ for all $e \in E$, where $S_{\mathcal{P}}$ and $T_{\mathcal{P}}$ are the sets of start and end vertices of the routes in $\mathcal{P}$, respectively. We quantify how well the reconstruction represents the flow field via the *flow deviation* $\Delta(\mathcal{P}, c, \phi)$, which we define as $\Delta(\mathcal{P}, c, \phi) = \sum_{e \in E} (\phi(e) - f_{(\mathcal{P},c)}(e))^2$. We say that a route is *realistic*, if the minimal *Fréchet distance* [3] to a trajectory $T \in \mathcal{T}$ is at most some parameter ε. We call a reconstruction *realistic* if all its routes are realistic.

Formal problem statement Given a road network $\mathcal{G} = (V, E)$, the representative trajectories $\mathcal{T}$, the flow field ϕ induced by the loop-detector data, and realism parameter $\varepsilon > 0$, compute a realistic reconstruction $(\mathcal{P}, c)$ such that the flow deviation $\Delta(\mathcal{P}, c, \phi)$ is minimized.

3 Route reconstruction algorithms

We describe two families of heuristics: Fréchet Routes (FR) which generate only realistic routes (Section 3.1) and variants of min-cost flow which relax the realism constraint (Section 3.2).

3.1 Fréchet Routes

To compute a reconstruction, we decouple finding a base set $\mathcal{P}$ and coefficients c. We do so iteratively, each time refining $\mathcal{P}$ and then computing the corresponding c. In each iteration, we perform three steps: (1) we generate candidate routes for each representative in $\mathcal{T}$ and add these to $\mathcal{P}$; (2) we compute the optimal c for the current base set, which can be done efficiently [5, 9]; (3) we prune $\mathcal{P}$ by eliminating duplicates and routes with coefficient zero.

The main question is how to generate candidate routes. To ensure realism, we modify the map-matching algorithm by Alt et al. [2] to generate a candidate for a trajectory $T \in \mathcal{T}$; see the full version for a short summary. This algorithm computes a path in a network $\mathcal{G}$ that is within Fréchet distance ε of T. Here we interpret T as a piece-wise linear function. A point $T(\tau)$ is *reachable* at a vertex v of $\mathcal{G}$, if there is a path in $\mathcal{G}$ ending at v, such that the Fréchet distance between this path and the subtrajectory up to $T(\tau)$ is at most ε. In the algorithm, the reachable points are represented as *reachable intervals* at each vertex: parts of the trajectory that have such a path ending at the vertex with Fréchet distance at most ε.

To achieve diversity and find routes that are likely to reduce the flow deviation, we present two extensions to this technique below. Both are steered by the *residual* flow field $\phi_r \colon E \to \mathbb{R}$, defined as $\phi_r(e) = \phi(e) - \sum_{P \in \mathcal{P}} M(P, e)c(P)$ for all edges $e \in E$. Specifically, we want to find candidate routes that currently have high residual flow.

Edge-inclusion Fréchet Routes (EFR) We modify the map-matching algorithm to find a route that must include a specific edge $e = (u, v)$ with high residual flow. To do so, we first construct a route from the start of T to u, finding all reachable intervals at u. Once these are found, we project these via e to reachable intervals at v and start a new search to construct a route from v to the end of T. We do this for up to k edges with the highest positive residual flow that are fully within the Minkowski sum of T with a disk of radius ε, for some parameter $k \geq 1$.

EFR stops its search in the free-space manifold as soon as the begin/endpoint of the reconstructed route lies within ε distance of the start/end of T. This might ignore flow on edges in the ε-vicinity of the start and end of T. Hence, we greedily add suitable edges to the ends of the route, taking care not to introduce cycles and not to decrease the average amount of residual flow per edge in the route.

Weighted Fréchet Routes (WFR) Contrasting EFR, we now aim to find a route with high total residual flow (weight). To this end, we refine each reachable interval into subintervals, based on a lower bound of the total weight by which it can be reached. This lower bound is simply the weight of the interval at u plus the residual flow of (u, v) when following the left-right pointers of an edge $(u, v) \in E$. Due to monotonicity, the subintervals increase in weight along the interval; the algorithm possibly prunes or shortens intervals found previously, if their lower bound was weaker. Cycles in the road network $\mathcal{G}$ can lead to cycles of dependency between the reachable intervals. To break such cyclic dependencies in deriving the candidate route, we construct a high-weight route explicitly via back-tracking, using each refinement at most once. Note that the resulting route may still contain cycles.

Hybrid (WEFR) We take the union of candidate routes generated by EFR and WFR.

3.2 Min-cost flow

We now describe two approaches that relax the realism constraint. For both, we solve a variant of the min-cost flow problem for ϕ and then decompose the result into a reconstruction.

Multi-commodity min-cost flow (MCMCF) For each representative trajectory T we construct a subgraph $G(T)$ of $\mathcal{G}$ consisting of all vertices and edges within distance ε of T. Vertices that are within distance ε from the start or end of T can act as sources or sinks of a flow in $G(T)$. Each representative trajectory hence induces a single (min-cost) flow problem. Using the graphs $G(T)$ for all $T \in \mathcal{T}$, we construct a *multi-commodity min-cost flow* problem on $\mathcal{G}$, where each trajectory T has an associated commodity which is restricted to $G(T)$. We can solve the resulting MCMCF using standard software packages (see Section 4).

Global min-cost flow (GMCF) We retain only the sources and sinks of MCMCF and otherwise impose no restriction on the flow. This results in a min-cost flow problem over the entire road network $\mathcal{G}$, which is essentially agnostic to the representative trajectories.

Heuristic path reconstruction Both approaches yield an edge flow (per commodity or overall). Our goal is to approximate these flows via a reconstruction that may use non-simple paths. For this, we first decompose the flow into path flows and cycle flows [1]. For every cycle flow, we add it to a path flow with which it shares a vertex; if such a path flow does not exist, we discard it. The resulting paths form the reconstruction.

4 Experimental Evaluation

We implemented all algorithms in C++ using Boost and MoveTK (https://movetk.win.tue.nl). For flow problems and determining coefficients c, we use IBM ILOG CPLEX 12.9. We ran all experiments single-threaded on an Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz.

Data Our road network $\mathcal{G}$ are the roads surrounding The Hague (the Netherlands) extracted from OpenStreetMap[1] with $n = 60\,277$ nodes and $m = 100\,654$ edges; see Figure 1. We derive ground-truth sets of routes $\mathcal{P}^*$ on $\mathcal{G}$ based on two complete sets of trajectories $\mathcal{T}^*$.

HS: $\mathcal{T}^* = \mathcal{P}^*$ consists of 5 000 shortest paths between random locations in $\mathcal{G}$; variation is created by randomly perturbing every edge length with a value in the interval $[0, \gamma]$, for some parameter $\gamma \geq 0$, separately for each shortest-path computation; we use $\gamma = 500m$.

HR: $\mathcal{T}^*$ is a set of 11 445 real-world trajectories in the The Hague area, provided by HERE Technologies. We map-match $\mathcal{T}^*$ to $\mathcal{G}$ to obtain $\mathcal{P}^*$. To avoid bias, we map-match using [16] instead of [2], as the latter is the basis for our Fréchet Routes.

We derive the flow field ϕ for $\mathcal{G}$ from $\mathcal{P}^*$. The representative trajectories $\mathcal{T}$ are a random sample of 5% of $\mathcal{T}^*$. For each dataset we create seven such samples.

Measures We quantify the performance via the five measures below (lower is better).

flow deviation flow induced by the reconstruction vs. input flow field: $\Delta(\mathcal{P}, c, \phi)$.

realism weighted average distance from reconstruction to ground truth: $\sum_{P \in \mathcal{P}} (c(P) \min_{T \in \mathcal{T}^*} d_F(P, T)) / \sum_{P \in \mathcal{P}} c(P)$.

coverage average distance from ground truth to reconstruction: $\text{avg}_{P^* \in \mathcal{P}^*} \min_{P \in \mathcal{P}} d_F(P^*, P)$.

complexity number of reconstructed routes (size of the basis): $|\mathcal{P}|$.

running time wall-clock time in minutes.

[1] Data © OpenStreetMap contributors; retrieved from `https://planet.osm.org/` in 2020.

Figure 1 Road network of The Hague, $|V| = 60\,277$, $|E| = 100\,654$ and its input flow fields.

Results We compare WEFR, MCMCF, and GMCF. Figure 2 visualizes the flow deviation as well as the ten reconstructed routes with the highest coefficient c. Though the overall deviation of GMCF (1.8×10^7) is lower than that of MCMCF (3.7×10^7) and WEFR (3.8×10^7), WEFR shows a spread of flow deviation over the network and often overrepresents ϕ, whereas GMCF and MCMCF have concentrated deviation along the major traffic axis and often underrepresent. We note that the most contributing routes for WEFR look like actual routes, whereas the other methods give fairly unintuitive routes: they typically are either very short (MCMCF) or follow an unrealistic path (GMCF).

We investigate the dependency on the realism threshold ε, using values $\{10\text{m}, 20\text{m}, 50\text{m}, 100\text{m}, 150\text{m}, 200\text{m}, 250\text{m}\}$. For WEFR we use $i_{\max} = 8$ and $k = 2$ (see full version for details).

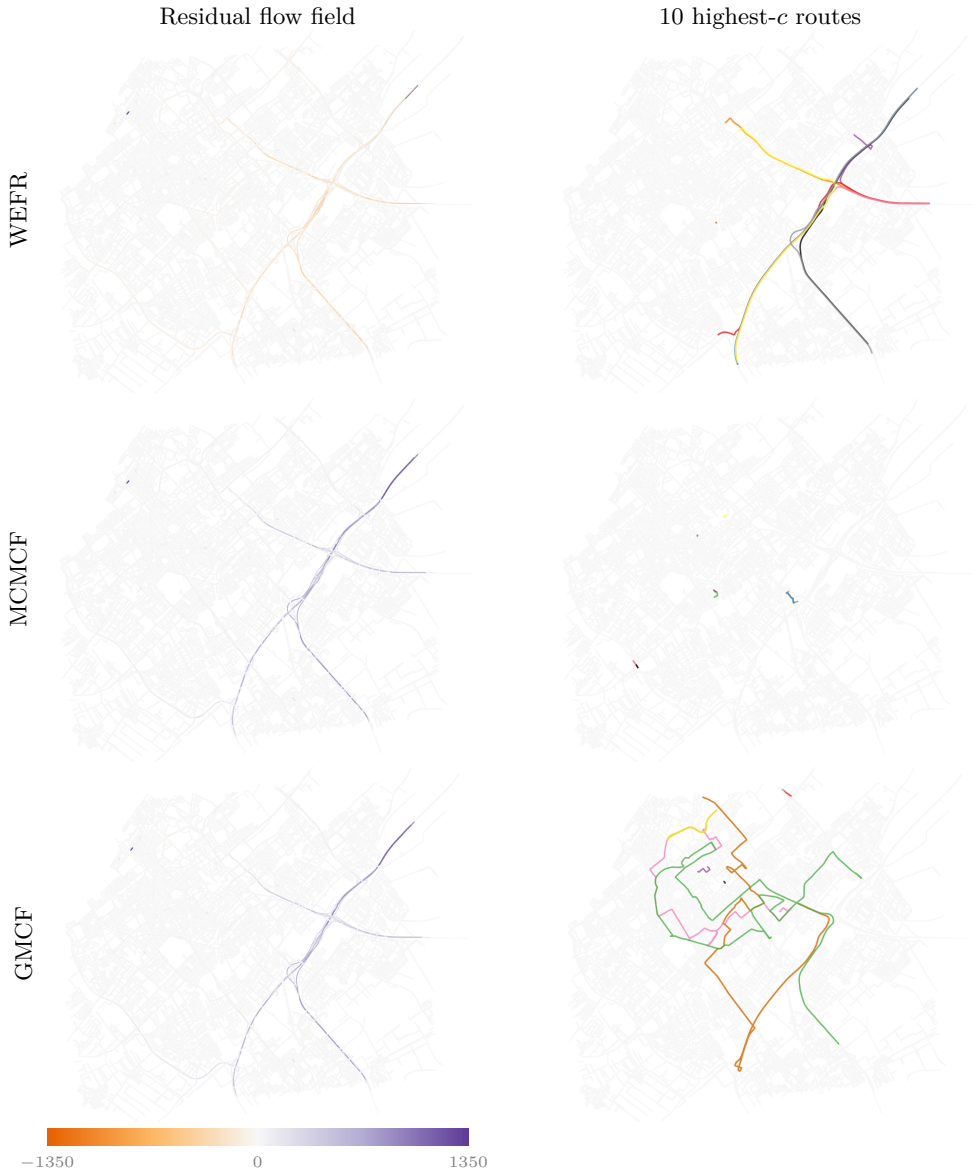


Figure 2 Residual flow field at the end of the algorithm (left) and the 10 highest-coefficient reconstructed routes (right) for one of the samples of HR for each technique with $\varepsilon = 100m$.

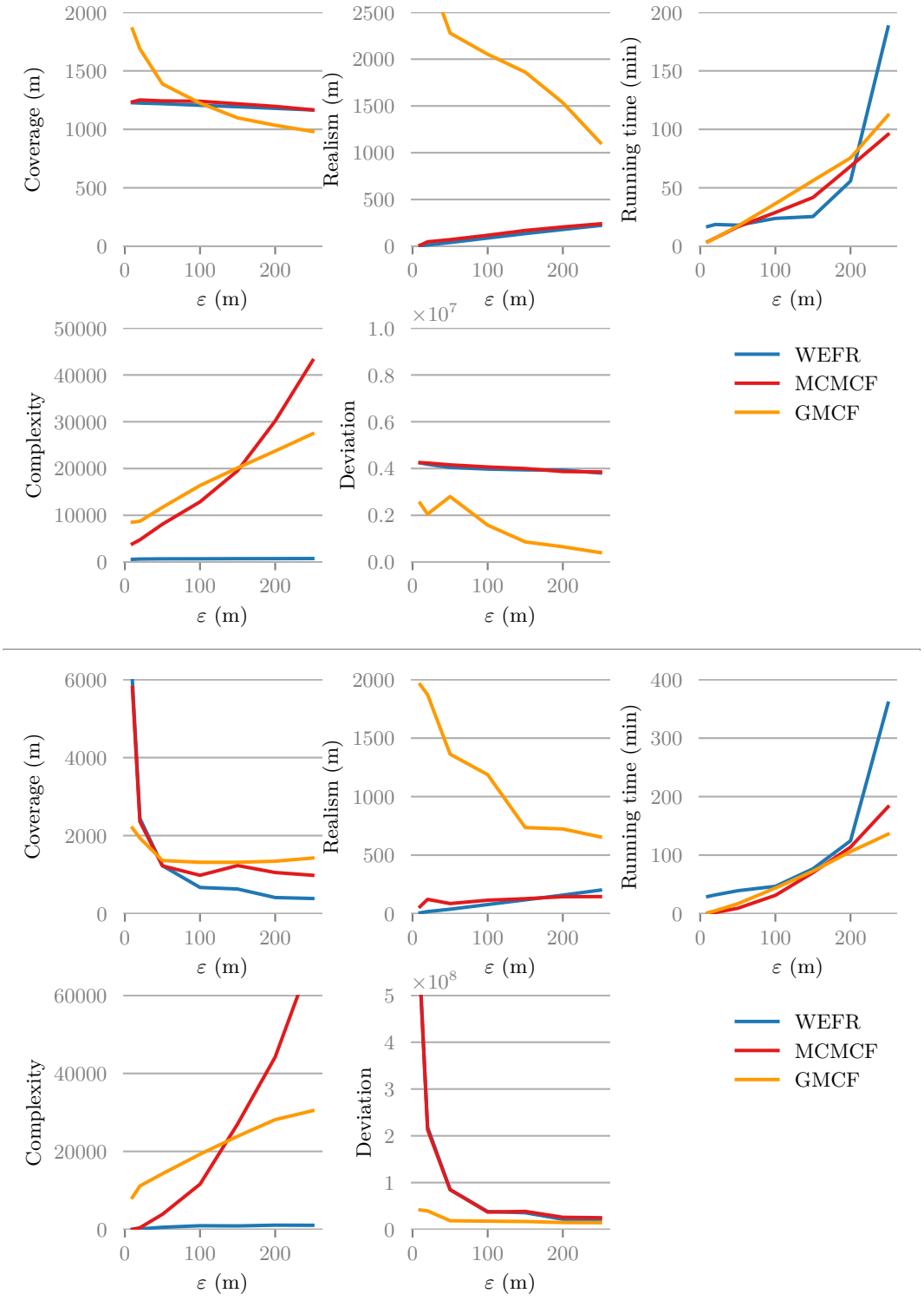


Figure 3 Results for our methods on HS (top) and HR (bottom) for varying ε.

We average the results over the seven samples per HS and HR. Since the complexity of MCMCF and GMCF is considerably larger than that of WEFR, we analyze realism and coverage for the 2 500 routes with highest coefficient (which is more than the complexity of WEFR). See Figure 3: generally the same patterns arise. Notably, realism improves with ε for GMCF, although it remains much worse than for WEFR and MCMCF. We attribute this to the flexibility offered by having more starting vertices in $\mathcal{G}$ available. MCMCF and WEFR behave similarly in coverage, realism and deviation. However, MCMCF has significantly higher complexity, whereas WEFR has higher running time for large ε. This suggests that realism is somewhat inherent in the flow information, demonstrating that the data sources are complementary. Finally, the pattern in coverage for GMCF and for WEFR and MCMCF seems to be opposite for the two datasets. We attribute this to the need for map-matching in HR which causes deviations between the flow field and the representative trajectories.

Representatives Our experiments show that increasing the distance threshold improves coverage and deviation, but reduces realism and efficiency. To mitigate these effects, we could preprocess using clustering and proceed with a set of central trajectories only, as very similar representatives do not offer much additional flexibility to the reconstruction. Furthermore, we could use the information from such clustering methods, or from map-matching accuracy, to vary the threshold ε per representative, to relate realism to the uncertainty in the data. We leave to future work to investigate how such techniques affect efficiency and quality.

Reconstruction Though MCMCF does not guarantee a bound on the Fréchet distance, it performs similarly to WEFR in realism, albeit with a very large basis. We could postprocess paths to allow only those that are realistic or attempt to incorporate realism into the path reconstruction phase of MCMCF, thereby reducing the complexity. Still, routes with a small Fréchet distance to the representatives can back-track for short distances, resulting in non-intuitive results. In future work we plan to consider stricter models for realism which still allow us to avoid the computational hardness associated with the simplicity requirement.

References

1 Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Pearson Education, 1993.

2 Helmut Alt, Alon Efrat, Günter Rote, and Carola Wenk. Matching planar maps. *Journal of Algorithms*, 49(2):262–283, 2003. doi:10.1016/S0196-6774(03)00085-3.

3 Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 5(1):75–91, 1995. doi:10.1142/S0218195995000064.

4 Quirijn W. Bouts, Irina Kostitsyna, Marc van Kreveld, Wouter Meulemans, Willem Sonke, and Kevin Verbeek. Mapping polygons to the grid with small Hausdorff and Fréchet distance. In *Proc. 24th European Symposium on Algorithms*, volume 57 of *LIPIcs*, pages 22:1–22:16, 2016. doi:10.4230/LIPIcs.ESA.2016.22.

5 Rasmus Bro and Sijmen De Jong. A fast non-negativity-constrained least squares algorithm. *Journal of Chemometrics*, 11(5):393–401, 1997. doi:10.1002/(SICI)1099-128X(199709/10)11:5<393::AID-CEM483>3.0.CO;2-L.

6 Tzvika Hartman, Avinatan Hassidim, Haim Kaplan, Danny Raz, and Michal Segalov. How to split a flow? In *Proc. IEEE International Conference on Computer Communications*, pages 828–836, 2012. doi:10.1109/INFCOM.2012.6195830.

7 Jan-Henrik Haunert and Benedikt Budig. An algorithm for map matching given incomplete road data. In *Proc. 20th International Conference on Advances in Geographic Information Systems*, pages 510–513, 2012. doi:10.1145/2424321.2424402.

8 George R. Jagadeesh and Thambipillai Srikanthan. Online map-matching of noisy and sparse location data with hidden Markov and route choice models. *IEEE Transactions on Intelligent Transportation Systems*, 18(9):2423–2434, 2017. doi:10.1109/TITS.2017.2647967.

9 Dongmin Kim, Suvrit Sra, and Inderjit S Dhillon. Tackling box-constrained optimization via a new projected quasi-Newton approach. *SIAM Journal on Scientific Computing*, 32(6):3548–3563, 2010. doi:10.1137/08073812X.

10 Kyle Kloster, Philipp Kuinke, Michael P. O'Brien, Felix Reidl, Fernando Sánchez Villaamil, Blair D. Sullivan, and Andrew van der Poel. A practical FPT algorithm for flow decomposition and transcript assembly. In *Proc. 20th Workshop on Algorithm Engineering and Experiments*, pages 75–86, 2018. doi:10.1137/1.9781611975055.7.

11 Maarten Löffler and Wouter Meulemans. Discretized approaches to schematization. In *Proc. 29th Canadian Conference on Computational Geometry*, 2017.

12 Paul Newson and John Krumm. Hidden Markov map matching through noise and sparseness. In *Proc. 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 336–343, 2009. doi:10.1145/1653771.1653818.

13 Mohammed A. Quddus, Washington Y. Ochieng, and Robert B. Noland. Current map-matching algorithms for transport applications: State-of-the art and future research directions. *Transportation Research Part C: Emerging Technologies*, 15(5):312–328, 2007. doi:10.1016/j.trc.2007.05.002.

14 Benedicte Vatinlen, Fabrice Chauvet, Philippe Chrétienne, and Philippe Mahey. Simple bounds and greedy algorithms for decomposing a flow into a minimal set of paths. *European Journal of Operational Research*, 185(3):1390–1401, 2008. doi:10.1016/j.ejor.2006.05.043.

15 Carola Wenk, Randall Salas, and Dieter Pfoser. Addressing the need for map-matching speed: Localizing global curve-matching algorithms. In *Proc. 18th International Conference on Scientific and Statistical Database Management*, pages 379–388, 2006. doi:10.1109/SSDBM.2006.11.

16 Can Yang and Gyozo Gidofalvi. Fast map matching, an algorithm integrating hidden Markov model with precomputation. *International Journal of Geographical Information Science*, 32(3):547–570, 2018. doi:10.1080/13658816.2017.1400548.

Route-preserving Road Network Generalization

Mees van de Kerkhof[1], Irina Kostitsyna[2], Marc van Kreveld[1], Maarten Löffler[1], and Tim Ophelders[2]

1 **Utrecht University**
{m.a.vandekerkhof, m.j.vankreveld, m.loffler}@uu.nl
2 **Eindhoven University of Technology**
{i.kostitsyna, t.a.e.ophelders}@tue.nl

Abstract

We investigate an approach for road network generalization, where the input is a road network and a collection of routes on these roads. The aim is to select a subset of the road network in which many routes of the collection are fully preserved. We investigate the complexity of this problem and show that it is NP-hard even when heavily restricted.

1 Introduction

Road network generalization is the process of reducing a road network in size by selecting only the most relevant roads. Data-driven methods rely on other data than just the road network. Car trajectories form the most obvious such other data source. Since route planning and traffic analysis often use a road map as an interface, the visualization and analysis of road networks are linked. If we want to support both use-cases simultaneously, we can use *route-preserving road network generalization (RPRNG)*, which we formulate as follows:

> Given a road network, represented by an embedded graph G, a set R of routes on this network, and a length budget B, compute a subgraph of G whose summed edge length is at most B and which contains the maximum number of routes of R in full.

This formulation has advantages over generalization that is not route-preserving: the subgraph that is obtained will represent as many driven routes as possible, with a preference for routes that are driven more often. The formulation is simple and intuitive, and the budget can be adjusted depending on the degree of generalization that is desired.

We want to emphasize the subtle difference between maximizing the total number of routes that are fully contained, and maximizing the total road usage of the roads chosen in the generalized network. In the latter problem, we can convert the routes into counts on the edges of the graph G, and otherwise forget about these routes. We believe that our version of preserving full routes gives rise to fewer artifacts. Figure 1 illustrates the difference and potential for artifacts. Maximum road usage leads to three "dead ends" in the generalized network that are not at destinations, and not a single route is fully preserved. Route-preserving generalization does not have dead ends, and two full routes are preserved.

1.1 Related work

There has already been a lot of research on what makes a good road network generalization. Among the criteria used for generalization, many are cartography-focused, in the sense that the generalized road network should "look good". These criteria include avoiding small faces between the roads, avoiding coalescence, and controlling density [2, 4, 8, 9, 11, 20, 23, 24]. Other methods give preference to sequences of roads that are smooth continuations of each

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

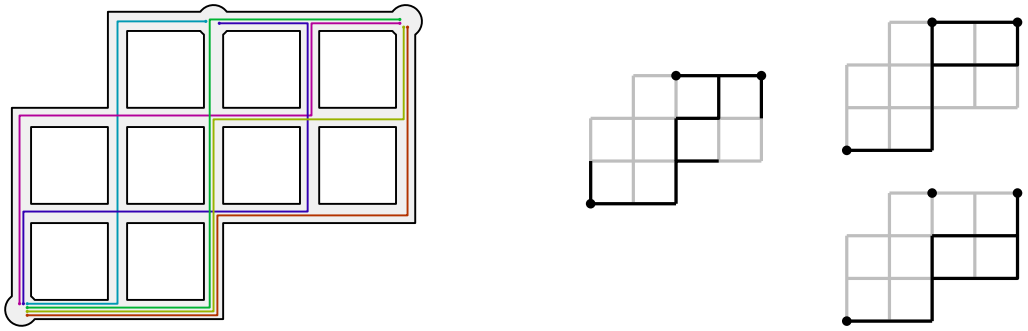


Figure 1 Left, a road network with six routes drawn. Vertices of the network that are the endpoint of a route are shown as disks. Middle, generalization according to maximum road usage with a budget of 11: all roads covered twice or more are chosen. Right, two equivalent outcomes of route-preserving generalization with a budget of 10: there are two ways to contain two routes with a budget of 10. A budget of 11 does not help to contain more routes.

other (strokes) [1, 10, 18, 16]. A global criterion that is often used is connectivity of the generalized network [3, 12], which is one of several structural graph-based criteria [6, 7, 21].

The most important criterion for continuous generalization is avoiding sudden changes, and avoiding that when zooming in one direction, a road segment disappears and then reappears [3, 15]. This type of behavior is to be avoided for all continuous generalization operations. One of the existing methods to road network generalization is coined "selective omission", where road segments, extended by good continuation to strokes, are scored based on geometric, topological, and attribute factors [2, 24]. Selecting roads based on how many routes are preserved can be used as another way for performing selective omission. Thus, a heuristical solver for the RPRNG problem can be adapted to support continuous zooming.

One of the first road network generalization methods can be called a precursor to data-driven generalization. Thomson and Richardson [17] let a subset of the vertices of a road network be sources and destinations, and they compute shortest paths between each pair. This gives artificial road usage, and they select the most used roads based on this computation. Similarly, road usage can be estimated based on an agent-based simulation [14]. While data-driven geography [13] has been around for longer, data-driven road network generalization has—to the best of our knowledge—not been studied until very recently. Fekete et al. [5] focus on finding the optimal placement for k points on a road network to maximize the length of real-data subtrajectories captured between each pair of points, which could also be applied for road network generalization. Yu et al. [22] refine road network generalization by selecting strokes in the network and incorporating traffic flows mined from trajectory data.

Besides the few other data-driven road network generalization papers, many methods use a scoring of roads, and clearly the road score can be based on actual road usage, which would be data driven (in both meanings of the word "driven"). Note that our route-preserving approach uses data in a different way: it is driven-route driven.

From the theoretical perspective, our research problem involves a weighted graph with paths on that graph. Our problem does not use coordinates of the road network explicitly (only to determine lengths of road segments), nor the coordinates and times of the trajectories. Hence, the abstract view of our problem is a graph problem and not a network problem. Since graph problems typically do not come with a set of paths on that graph, there are no closely related graph problems. However, there are some connections which allow us to prove NP-hardness of route-preserving road network generalization.

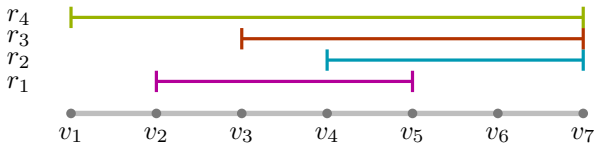


Figure 2 Example of the RPRNG problem on a path graph with seven vertices and four routes. An optimal solution to cover two routes is a path between v_3 and v_7, which covers routes r_2 and r_3.

2 Theoretical results

In this section we show that the RPRNG problem is NP-hard, even for simple, sparsely covered road networks. We do show the problem can be solved quickly in two special cases: We present a polynomial-time algorithm for when the road network is a path, and an FPT algorithm for when it is a tree with each segment of the network covered by at most c routes, for some fixed constant c.

Let $G = (V, E)$ be the graph representing the road network, with vertices V and edges E, and let R be the set of routes in G. For simplicity of presentation, we assume that each route in R begins and ends at a vertex in V, and that the routes are simple paths (i.e., non-self-intersecting). Furthermore, assume that each edge in G is traversed by at least one route, otherwise we remove such edges in a preprocessing step. Define the *ply* of an edge $e \in E$ to be the number of routes in R traversing e. Similarly, the *ply* of a vertex $v \in V$ is the number of routes in R traversing v, not including routes ending at v.

Our theoretical results are summarized in Table 1. For our hardness proofs we reduce to the decision version of the problem. For space reasons only sketches of the proofs are given.

2.1 When G is a path

We start with a dynamic programming (DP) algorithm for the case when $G = (V, E)$ is a path. Let $V = \{v_1, v_2, \ldots, v_n\}$ be n vertices of G embedded on a line from left to right, and let edges $E = \{e_i = (v_i, v_{i+1}) \mid 1 \leq i < n\}$ (refer to Fig. 2). Let R be a set of m routes on G, and let R_i be the set of routes that each start on or before v_i and end after v_i. Let $F(i, k, S)$, where $i \leq n$, $k \leq m$, and $S \subset R_i$, denote the solution for a subproblem in our dynamic programming formulation on the first i vertices, with at least k routes to be covered, and such that the routes in S are all covered up to vertex v_i. The goal is to minimize the total length of an output subgraph in $F(i, k, S)$. A naive DP is given by the following recurrence:

$$F(i, k, S) = \min_{S \cap R_{i-1} \subseteq S' \subseteq R_{i-1}} \left[F(i-1,\, k - |S' \setminus R_i|,\, S') + cost(e_{i-1}, S') \right],$$

Table 1 Algorithmic and hardness results for graph classes of G and bounds on the ply.

	path	tree	planar graph
ply 1	poly-time	poly-time	poly-time
ply 2	poly-time	poly-time	NP-hard
ply c	poly-time	FPT	NP-hard
ply ∞	poly-time	NP-hard	NP-hard

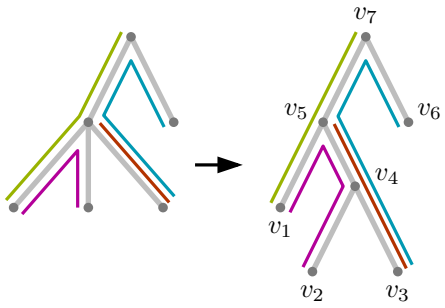


Figure 3 Example of the RPRNG problem on a tree graph with four routes. Vertices of degree higher than three are split to form a binary tree, weight of edge (v_4, v_5) is set to 0.

with boundary conditions $F(i, k, S) = 0$ if $i = 0$ and $k \leq 0$, and $F(i, k, S) = \infty$ if $i = 0$ and $k > 0$. We define the cost function

$$cost(e, S') = \begin{cases} 0\,, & \text{if } S' = \varnothing\,, \\ \|e\|\,, & \text{otherwise}\,. \end{cases}$$

So the length of an edge e is included in the total cost if and only if it is covered by one of the selected trajectories. Note that the recursion as written considers exponentially many subsets S'. The path constraint lets us optimize this as we can compute and consider only the $O(|R|)$ maximal subsets instead. This means we have a polynomial size DP table and can solve subproblems in $O(|R|)$ time, giving the following theorem.

▶ **Theorem 2.1.** *The RPRNG problem can be solved in $O(n|R|^3)$ time if G is a path.*

2.2 When G is a tree with bounded ply

We now modify our dynamic programming algorithm to solve the RPRNG problem when G is a tree, and the maximum ply is bounded by some constant c. The vertices of degree higher than three can be split by inserting zero-weight edges to form a binary tree. Choose an arbitrary root, and order the vertices according to the post-order traversal (refer to Fig. 3). Let R_i be the set of routes which start on or below v_i and traverse v_i from bottom to top.

Consider a subproblem $F(i, k, S)$ on a subtree rooted at the vertex v_i, where at least k routes must be covered, including the set $S \subset R_i$. In our dynamic program, we will merge the solutions of the subproblems defined on the subtrees rooted at the children of v_i. The DP recursion is given by the following formula:

$$F(i,k,S) = \min_{\substack{S\cap R_\ell \subseteq S_\ell \subseteq R_\ell \\ S\cap R_r \subseteq S_r \subseteq R_r \\ k_\ell + k_r = k - |S_\ell \cap S_r|}} \left[F(i_\ell, k_\ell, S_\ell) + cost(e_{i_\ell}, S_\ell) \; + F(i_r, k_r, S_r) + cost(e_{i_r}, S_r)\right],$$

where i_ℓ and i_r index the left and right child of the vertex v_i respectively, and e_{i_ℓ} and e_{i_r} denote the edges (v_{i_ℓ}, v_i) and (v_{i_r}, v_i).

The boundary conditions are $F(i, k, S) = 0$ if v_i is a leaf and $k \leq 0$, and $F(i, k, S) = \infty$ if v_i is a leaf and $k > 0$. The values k_ℓ and k_r in the two subproblems depend on the value k and the number of routes covered by the solutions of the two subproblems which are traversing v_i from the left subtree into the right subtree (for example, for the vertex v_5 in the Fig. 3, the purple route is such route). More specifically, let k' be the number of routes in $S_\ell \cap S_r$, then $k = k' + k_\ell + k_r$.

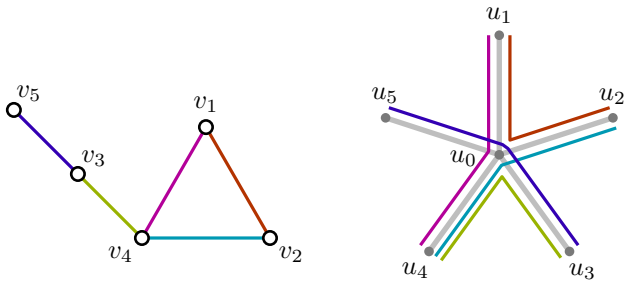


Figure 4 Left: graph G' from a CLIQUE instance. Right: graph G with five routes corresponding to the edges of G'. Subgraph of G on vertices $\{u_0, u_1, u_2, u_4\}$ corresponds to the clique $\{v_1, v_2, v_4\}$.

Unlike in the previous section, we can no longer consider only maximal subsets S of the routes traversing v_i; we need to consider all possible subsets. The size of the dynamic programming table becomes $n \times |R| \times 2^c = O(n|R|)$, and we spend $O(|R|2^c) = O(|R|)$ time per subproblem. We conclude with the following theorem.

▶ **Theorem 2.2.** *The RPRNG problem, when G is a tree, is fixed-parameter tractable with the maximum ply as the parameter: If the maximum ply is bounded by some constant c, it can be solved in $O(n|R|^2)$ time.*

2.3 When G is a tree with unbounded ply

In this section we show that the RPRNG problem is NP-hard by a reduction from the CLIQUE problem. Given an instance of the CLIQUE problem on a graph G' with sought clique of size k', we construct an instance of the RPRNG problem consisting of a road network graph G, a set of routes R, an integer budget B, and an integer k and show that there is a clique of size at least k' in G' if and only if there is a subgraph of G of total length at most B which completely covers at least k routes from R.

We do this by creating a star graph, where there is a leaf vertex for each vertex of the original graph, connected by an edge to a central vertex. For each edge of the original graph there is a route on the star graph running between the vertices associated with the endpoints of the edge; see Figure 4. We set B to k' and k to $k'(k'-1)/2$. Because a clique of n vertices has $n(n-1)/2$ edges, this means we can only capture k of the routes if and only if G' has a clique of size k'.

▶ **Theorem 2.3.** *The RPRNG problem with the objective to maximize the number of covered routes is NP-hard when G is a tree and the maximum ply is unbounded.*

2.4 When G is a planar graph with bounded ply

Finally, we show that the RPRNG problem is NP-hard when G is a planar graph even if the ply on edges and vertices is bounded by 2. We again reduce from the CLIQUE problem.

Consider an instance of the CLIQUE problem on a graph $G' = (V', E')$, with an integer k. We construct an instance of the RPRNG problem consisting of a graph G, a set of routes R, a budget B, and the same integer k, such that G' has a clique of size k if and only if there exists a subgraph of G of total weight at most B which covers at least k routes.

To create this instance, we set G to be a $|V'| \times 2|E'|$ grid graph, where we assign one row to each vertex of V'. We assign two adjacent columns of vertices to each edge of E',

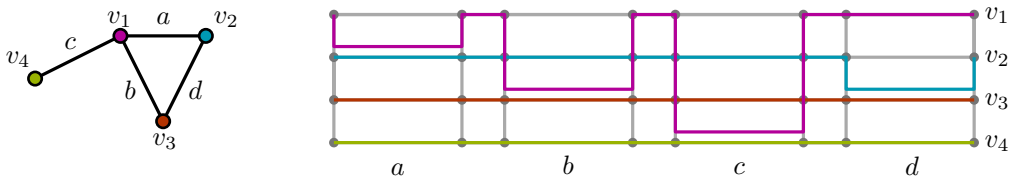


Figure 5 Left: graph G' of a CLIQUE instance. Right: corresponding grid graph G, and four routes corresponding to the vertices of G'. Edges in a column associated with an edge of G' have weight 1, all other edges have weight 0

and set the weight of the $|V'|$ horizontal edges connecting these adjacent columns to 1. All other edges have weight 0. We create a route for each vertex of V' that lies on the row associated with its vertex, except for the edges in columns associated with edges where the associated vertex is the lower-indexed endpoint. For those edges, the route travels vertically to the row associated with the higher-indexed endpoint and uses that edge instead before vertically going back to its own row. See Figure 5. We set $B = k|E| - k(k-1)/2$. That this is a correct reduction becomes clear when we consider that if we pick k routes that do not overlap, we need a budget of $k|E|$. If two selected routes overlap, the amount of budget needed decreases by 1. If the savings total $k(k-1)/2$, this means we can select k vertices that share $k(k-1)/2$ edges between them in G', implying they are a clique of size k. It is easy to see that in the grid graph, no edge or vertex is covered by more than two routes.

▶ **Theorem 2.4.** *The RPRNG problem with the objective to maximize the number of covered routes is NP-hard even if G is a planar graph and the maximum ply is bounded by* 2.

3 Conclusion

We have introduced a new, data-driven approach to road network generalization that can be used when trajectory data is available as well. Theoretical analysis shows that the problem is NP-hard for almost all variants. This paper presents our theoretical results. To assess and validate the concept of route-preserving road network generalization, we study heuristic approaches for solving this problem in [19]. An image of a generalization generated by one of the heuristics can be seen in Figure 6. Theoretical future work can be done on finding an algorithm with a proven approximation factor in the cases where the problem is NP-hard.

Acknowledgments. This research was started at the AGA workshop, January 2020, in Langbroek (NL). M.v.d.K., M.v.K., and M.L. are (partially) supported by the Dutch Research Council on the Commit2Data project "Geometric Algorithms for the Analysis and Visualization of Heterogeneous Spatio-temporal Data" (no. 628.011.005). M.L. is partially supported by Dutch Research Council on grant no. 614.001.504. The authors thank HERE Technologies for providing data.

References

1 Stefan A. Benz and Robert Weibel. Road network selection for medium scales using an extended stroke-mesh combination algorithm. *Cartography and Geographic Information Science*, 41(4):323–339, 2014.

2 Jun Chen, Yungang Hu, Zhilin Li, Renliang Zhao, and Liqiu Meng. Selective omission of road features based on mesh density for automatic map generalization. *International Journal of Geographical Information Science*, 23(8):1013–1032, 2009.

Figure 6 Top: Road network of the city of Leiden. Bottom: Heuristic generalization restricted to 30% of the total road length.

3 Markus Chimani, Thomas C. van Dijk, and Jan-Henrik Haunert. How to eat a graph: Computing selection sequences for the continuous generalization of road networks. In *Proc. 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 243–252, 2014.

4 Cécile Duchêne, Blanca Baella, Cynthia A Brewer, Dirk Burghardt, Barbara P. Buttenfield, Julien Gaffuri, Dominik Käuferle, François Lecordix, Emmanuel Maugeais, Ron Nijhuis, Maria Pla, Marc Post, Nicolas Regnauld, Lawrence V. Stanislawski, Jantien Stoter, Katalin Tóth, Sabine Urbanke, Vincent van Altenaand, and Antje Wiedemann. Generalisation in practice within national mapping agencies. In D. Burghardt et al., editor, *Abstracting Geographic Information in a Data Rich World*, pages 329–391. Springer, 2014.

5 Sándor P. Fekete, Alexander Hill, Dominik Krupke, Tyler Mayer, Joseph S. B. Mitchell, Ojas Parekh, and Cynthia A. Phillips. Probing a Set of Trajectories to Maximize Captured Information. In *18th International Symposium on Experimental Algorithms (SEA 2020)*, pages 5:1–5:14, 2020.

6 Bin Jiang and Christophe Claramunt. A structural approach to the model generalization of an urban street network. *GeoInformatica*, 8(2):157–171, 2004.

7 Bin Jiang and Christophe Claramunt. Topological analysis of urban street networks. *Environment and Planning B: Planning and design*, 31(1):151–162, 2004.

8 Bin Jiang and Lars Harrie. Selection of streets from a network using self-organizing maps. *Transactions in GIS*, 8(3):335–350, 2004.

9 Xingjian Liu, Tinghua Ai, and Yaolin Liu. Road density analysis based on skeleton partitioning for road generalization. *Geo-spatial Information Science*, 12(2):110–116, 2009.

10 Xingjian Liu, F Benjamin Zhan, and Tinghua Ai. Road selection based on Voronoi diagrams and "strokes" in map generalization. *International Journal of Applied Earth Observation and Geoinformation*, 12:S194–S202, 2010.

11 William Mackaness. Analysis of urban road networks to support cartographic generalization. *Cartography and Geographic Information Systems*, 22(4):306–316, 1995.

12 William A. Mackaness and Kate M. Beard. Use of graph theory to support map generalization. *Cartography and Geographic Information Systems*, 20(4):210–221, 1993.

13 Harvey J. Miller and Michael F. Goodchild. Data-driven geography. *GeoJournal*, 80(4):449–461, 2015.

14 B Morisset and Anne Ruas. Simulation and agent modelling for road selection in generalisation. In *Proc. ICA 18th International Cartographic Conference*, pages 1376–1380, 1997.

15 Radan Šuba, Martijn Meijers, and Peter Van Oosterom. Continuous road network generalization throughout all scales. *ISPRS International Journal of Geo-Information*, 5(8):145, 2016.

16 Robert C. Thomson and Rupert Brooks. Exploiting perceptual grouping for map analysis, understanding and generalization: The case of road and river networks. In *International Workshop on Graphics Recognition*, pages 148–157, 2001.

17 Robert C. Thomson and Dianne E. Richardson. A graph theory approach to road network generalisation. In *Proc. 17th International Cartographic Conference*, pages 1871–1880, 1995.

18 Robert C. Thomson and Dianne E. Richardson. The 'good continuation' principle of perceptual organization applied to the generalization of road networks. In *Proc. of the 19th International Cartographic Conference*, pages 1215—-1223, 1999.

19 Mees van de Kerkhof, Irina Kostitsyna, Marc van Kreveld, Maarten Löffler, and Tim Ophelders. Route-preserving road network generalization. In *Proceedings of the 28th International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '20, page 381–384, 2020.

20 Marc van Kreveld and Jarno Peschier. On the automated generalization of road network maps. In *Proc. 3rd International Conference in GeoComputation*, 1998.

21 Roy Weiss and Robert Weibel. Road network selection for small-scale maps using an improved centrality-based algorithm. *Journal of Spatial Information Science*, 2014(9):71–99, 2014.

22 Wenhao Yu, Yifan Zhang, Tinghua Ai, Qingfeng Guan, Zhanlong Chen, and Haixia Li. Road network generalization considering traffic flow patterns. *International Journal of Geographical Information Science*, 34(1):119–149, 2020.

23 Qingnian Zhang. Modeling structure and patterns in road network generalization. In *ICA Workshop on Generalisation and Multiple Representation*, 2004.

24 Qi Zhou and Zhilin Li. Evaluation of properties to determine the importance of individual roads for map generalization. In *Advances in Cartography and GIScience*, volume 1, pages 459–475. Springer, 2011.

Path-Greedy Spanner in Near-Quadratic Time: Simpler and Better

Paz Carmi[1] **and Idan Tomer**[2]

1 **Ben-Gurion University of the Negev**
carmip@bgu.ac.il
2 **Ben-Gurion University of the Negev**
idantom@post.bgu.ac.il

Abstract

The Path-Greedy algorithm produces high-quality spanners, having the most desirable properties of geometric spanners both in theory and in practice. More specifically, it has a natural definition, small degree, linear number of edges, low weight, and strong t-spanner for every $t > 1$.

In this paper we revisit the Path-Greedy spanner and present an algorithm that computes the Path-Greedy spanner in $O(n^2 \log n)$ time. This algorithm is based on Bose et al. $O(n^2 \log n)$ time algorithm, however, our algorithm is easier to implement, and in practice is potentially more efficient.

1 Introduction

Geometric spanners are fundamental structures that have attracted a great deal of research attention in the past few decades. Given a weighted graph G and a real number $t > 1$, a t-spanner of G is a spanning sub-graph G^* with the property that for every edge $(p, q) \in G$ there exists a path between p and q in G, whose weight is no more than t times the weight of the edge (p, q). Thus, shortest-path distances in G^* approximate shortest-path distances in the underlying graph G, and the parameter t represents the approximation ratio. The smallest t for which G^* is a t-spanner of G is known as the spanning ratio of the graph G^*.

Spanners have been studied in many different settings, which depend on different aspects, such as the type of underlying graph G, on the way in which weights are assigned to edges in G, the specific value of the spanning ratio t, and on the function used to measure the weight of a shortest path. In this paper, we concentrate on the setting where the underlying graph is the complete geometric graph over set of points P, and a weight of an edge (p, q) is equal to the distance $d(p, q)$ between its endpoints p and q, such that (P, d) is a finite metric space.

Probably one of the most studied geometric spanner is the *Path-Greedy* spanner introduced by Althöfer et al. [3], see Algorithm 1. The Path-Greedy algorithm produces a t-spanning graph with linear number of edges, bounded degree and bounded weight. The main disadvantage of the Path-Greedy algorithm is its high time complexity, which is $O(n^3 \log n)$. Therefore, many attempts were made to reduce the construction time and in parallel to compute more efficiently other types of geometric spanners.

Various experiments showed that the Path-Greedy algorithm produces spanners whose size, weight, and maximum degree are much lower than the spanners produced by other approaches. In [7], Das and Narasimhan showed how to construct a spanner that approximates the Path-Greedy spanner in $O(n \log^2 n / \log \log n)$ time. This spanner is known in the literature as the *approximate-Greedy*. In [8], Farshi and Gudmundsson showed that while the theoretical bounds of the approximate-Greedy spanner is similar to the original Path-Greedy spanner with respect to the number of edges, the degree, and the weight of the graph, in

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Algorithm 1 Path-Greedy(P, t)

Input: A set P of points in the plane and a constant $t > 1$
Output: A t-spanner $G = (P, E)$ for P

```
L ← the (n choose 2) pairs of distinct points sorted in non-decreasing order of their distances
E ← ∅
for all (p, q) ∈ L do
    π ← length of the shortest path in G between p and q
    if π > t · d(p, q) then
        E ← E ∪ {(p, q)}
    end if
end for
return G = (P, E)
```

practice the approximate-Greedy spanner behaves significantly worse with respect to these properties. In [5], Bose et al. showed an algorithm that computes the Path-Greedy spanner in $O(n^2 \log n)$ time. Their algorithm maintains a stack for each point to be able to restore the Dijkstra algorithm computation. In [1, 2, 6], the authors also considered simplifying the Path-Greedy, and introduced algorithms that are more efficient in practice, e.g., linear in space.

In [4], Bar-On and Carmi introduced the *δ-Greedy t-spanner* that has the same theoretical and practical properties as the Path-Greedy spanner. They showed a simple algorithm that computes δ-Greedy t-spanner in $O(n^2 \log n)$ time.

In this paper we suggest a simplification to Bose et al. [5] algorithm construction. Our algorithm is easier for understanding and implementation. In addition, to implement our algorithm we require to maintain simple data structures. Moreover, even though both algorithms have the same theoretical running time, our algorithm in practice is potentially more efficient.

2 A Simple Near-Quadratic Greedy Algorithm

In this section, we show how to simplify the algorithm presented by Bose et al. [5], which computes the Path-Greedy spanner in near-quadratic time. The main difference is that in our algorithm there is no need to maintain a stack for the Dijkstra re-computation for each point. Notice that the hardest part to implement in Bose et al. [5] algorithm is the stacks, that are needed to maintain the required information for performing the undo operations on the Dijkstra computations.

The high-level description of our algorithm is similar to Algorithm 3.2 introduced by Bose et al. in [5], as presented in Section 2.1

2.1 Algorithm description

The high-level description of Algorithm 2 is as follows:

(i) For each point in P, initialize an empty Dijkstra's priority queue, and for each pair (u, v), set its entry in the weight-matrix with the value $wt(u, v) = \infty$ if $u \neq v$, and $wt(u, v) = 0$ otherwise.

(ii) Sort the $\binom{n}{2}$ pairs of distinct points in P in non-decreasing order of their distances, and partition them into buckets, such that bucket i contains the set of pairs E_i of distance between L_i and $2L_i$.

(iii) Process the buckets in the sorted order, and for each bucket i:

- for each point in P, run *Bounded-Dijkstra* (*SSSP*) up to $2tL_i$. More precisely, for each point $p \in P$, run Dijkstra until the key in p's priority queue is at most $2tL_i$; and
- for each pair $(u, v) \in E_i$ in sorted order, if $wt(u, v) > t \cdot d(u, v)$, then add (u, v) to E and make local updates in the weight-matrix, such that all entries that correspond to pairs in E_i are equal to the shortest-path distance in the updated graph G (i.e., update the weight of all points that are not too "far away" from u or v).

In Algorithm 3.2 [5], after adding the edge (u, v), all the points that are in the vicinity of at least one of the endpoints u or v are updated. The update is performed by executing *Dijkstra-Undo* and then *Bounded-Dijkstra* again, while updating the priority queue and the weight-matrix for each updated point. The *Dijkstra-Undo* runs Dijkstra's algorithm backwards as long as the minimum key in the priority queue PQ_p of p is at least $\min\{(t - 1/2)Li, Li)\}$. In order to perform the *Dijkstra-Undo*, the algorithm maintains a stack τ_p for each $p \in P$ containing all the operations of the SSSP from p. In our algorithm, we find the points that lie in the vicinity of u (and similarly of v), by traversing all the points, such that their distance from u is between 0 to $2tL_i - d(u, v)$, scanning in increasing distance. For each such point p (i.e., $wt(p, u) \leq 2tL_i - d(u, v)$), we traverse all the points q, such that $wt(v, q) \leq 2tL_i - [wt(p, u) + d(u, v)]$; see Figure 1. Then, for each such q, update $wt(p, q)$ and $wt(q, p)$ in the weight-matrix to the minimum between its current value and $wt(p, u) + d(u, v) + wt(v, q)$.

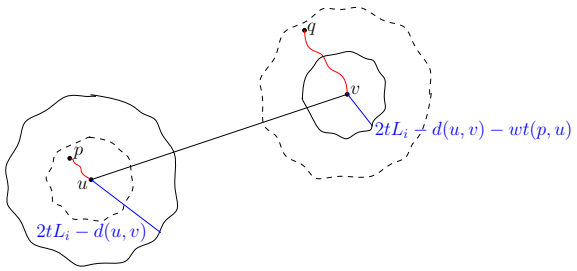


Figure 1 Updating the points in the vicinity of an added edge (u, v).

In Algorithm 3.2 [5] (Line 20), after processing all the edges of bucket i, and before proceeding to next bucket, the algorithm performs *Dijkstra-Undo* to update all keys in the priority queues of all the points which may be affected by the addition of the edges of bucket i to the resulted spanner. In our algorithm, to update these keys in the priority queues, we run Procedure 3, before proceeding to the next bucket. In this procedure, for every edge (u, v) in bucket i that was added to the resulted graph, we update the priority queue PQ_p for all the points $p \in P$, such that $wt(p, u) < 2tL_i$ or $wt(p, v) < 2tL_i$. That is, we update the key of u (resp., of v) in the priority queue of p to be the minimum between the current value and $wt(p, u) + d(u, v)$ (resp., $wt(p, v) + d(u, v)$). Similarly, we update the key of p in the priority queue of u and in the priority queue of v.

To summarize, we show that one can use only local information that already exists in the weight-matrix to update the distances between pairs of points that can be affected from the addition of a new edge to the spanner. Actually, this modification results in less computations required per point, since here we update only the relevant points while in [5] this is not the case.

Algorithm 2 Simplified-Path-Greedy(P, t)

Input: A set P of points in the plane and a constant $t > 1$
Output: the greedy t-spanner $G(P, E)$

for all $u, v \in P$ **do**
  $wt(u, v) \leftarrow 0$
  if $u \neq v$ **then**
    $wt(u, v) \leftarrow \infty$
  end if
end for
$E' \leftarrow \binom{n}{2}$ pairs of distinct points, sorted in non-decreasing order of their distances
$i \leftarrow 1$, $E_0 \leftarrow \emptyset$
while $E' \setminus (\bigcup_{k=0}^{i-1} E_k) \neq \emptyset$ **do**
  $L_i \leftarrow$ distance of shortest pair in $E' \setminus (\bigcup_{k=0}^{i-1} E_k)$
  $E_i \leftarrow$ sorted list of all pairs in $E' \setminus (\bigcup_{k=0}^{i-1} E_k)$ whose distances are in $[L_i, 2L_i)$
  $i \leftarrow i + 1$
end while
$l \leftarrow i - 1$
$E \leftarrow \emptyset$
$G \leftarrow (P, E)$
for all $u \in P$ **do**
  $PQ_u \leftarrow$ priority queue storing all $v \in P$ with the key $wt(u, v)$
  $SW_u \leftarrow$ data structure storing all $v \in P$ in sorted order by $wt(u, v)$
end for
for $i \leftarrow 1, \ldots, l$ **do**
  for all $u \in P$ **do**
    Bounded-Dijkstra$(G, u, 2tL_i, PQ_u)$
  end for
  $E_N \leftarrow \emptyset$
  for all $(u, v) \in E_i$ (in sorted order) **do**
    if $wt(u, v) > t \cdot d(u, v)$ **then**
      $E \leftarrow E \cup \{(u, v)\}$
      $E_N \leftarrow E_N \cup \{(u, v)\}$
      for all $p \in SW_u$ s.t $wt(p, u) \leq 2tL_i - d(u, v)$ (in sorted order) **do**
        for all $q \in SW_v$ s.t $wt(v, q) \leq 2tL_i - [wt(p, u) + d(u, v)]$ (in sorted order) **do**
          $d \leftarrow wt(p, u) + d(u, v) + wt(v, q)$
          $wt(p, q) \leftarrow \min(wt(p, q), d)$
          $wt(q, p) \leftarrow wt(p, q)$
          update SW_p and SW_q
        end for
      end for
    end if
  end for
  Update(E_N)
end for
return G

Procedure 3 Update(E_N)

Input: A set E_N of new edges added in each iteration in Algorithm 2

1: **for each** $(u, v) \in E_N$ **do**
2: **for each** $p \in SW_u$ such that $wt(p, u) < 2tL_i$ **do** /* in sorted order */
3: $\delta \leftarrow wt(p, u) + d(u, v)$
4: decrease the key of v in PQ_p to δ (if $key > \delta$)
5: decrease the key of p in PQ_v to δ (if $key > \delta$)
6: update SW_p and SW_u
7: **for each** $p \in SW_v$ such that $wt(p, v) < 2tL_i$ **do** /* in sorted order */
8: $\delta \leftarrow wt(p, v) + d(u, v)$
9: decrease the key of u in PQ_p to δ (if $key > \delta$)
10: decrease the key of p in PQ_u to δ (if $key > \delta$)
11: update SW_p and SW_v

2.2 Correctness and running time

The correctness of our algorithm follows directly from the correctness of Algorithm 3.2 [5], since they both perform the same required updates after adding an edge.

Next, we show that the running time of our algorithm is asymptotically similar to Algorithm 3.2 [5]. There are two differences between our algorithm and Algorithm 3.2 [5] with respect to the running time. The first difference is that Algorithm 3.2 [5] preforms *Dijkstra-Undo* and *Bounded-Dijkstra*, for every point that satisfies the update condition (see Line 28 in Algorithm 3.2 [5]), and our algorithm scans all the points in increasing order, and then, for each pair that satisfies our condition (see Lines 29-30 in Algorithm 2), it updates the points' weights in $O(1)$ time, and its SW in $O(\log n)$ time (using data structure e.g., AVL or Skip-List). Observe that the number of pairs examined by our algorithm is a subset of the pairs examined by the *Bounded-Dijkstra* executed in Algorithm 3.2 [5]. The second difference is that our algorithm, updates the priority queue, PQ_p in the end of each bucket for all $p \in P$, such that $d(p, u) < 2tL_i$ or $d(p, v) < 2tL_i$ for all (u, v) that was added to the spanner in this bucket. To go through the points in increasing order, we use a data structure for SW (e.g., AVL or Skip-List). To maintain the priority queue, we update each point in $O(\log n)$ time (even though implementations with $O(1)$ time exists, for simplicity, and since this part does not affect the overall running time, we use $O(\log n)$ updating time). Finally, for every $p \in P$, the number of times p can be included in such update process is bounded by $O(1/(t-1)^{O(d)})$ for every bucket (as shown in Lemma 3 [5]). Maintaining the queues and SW data structures takes $O(1/(t-1)^{O(d)} n \log n)$ time, since the number of buckets is $O(n)$. Executing *Bounded-Dijkstra* in the beginning of each bucket takes $O(n \log n)$ time, and thus, the total running time of our algorithm is $O(1/(t-1)^{O(d)} n^2 \log n)$.

References

1 S. P. A. Alewijnse, Q. W. Bouts, Alex P. ten Brink, and K. Buchin. Distribution-sensitive construction of the greedy spanner. *Algorithmica*, pages 1–23, 2016.

2 Sander P. A. Alewijnse, Quirijn W. Bouts, Alex P. ten Brink, and Kevin Buchin. Computing the greedy spanner in linear space. *Algorithmica*, 73(3):589–606, 2015.

3 I. Althöfer, G. Das, D. Dobkin, D. Joseph, and J. Soares. On sparse spanners of weighted graphs. *Discrete and Computational Geometry*, 9(1):81–100, 1993.

4 Gali Bar-On and Paz Carmi. δ-greedy t-spanner. In *WADS*, pages 85–96, 2017.

5 Prosenjit Bose, Paz Carmi, Mohammad Farshi, Anil Maheshwari, and Michiel H. M. Smid. Computing the greedy spanner in near-quadratic time. *Algorithmica*, 58(3):711–729, 2010.

6 Quirijn W. Bouts, Alex P. ten Brink, and Kevin Buchin. A framework for computing the greedy spanner. In *SOCG'14*, page 11, 2014.

7 G. Das and G. Narasimhan. A fast algorithm for constructing sparse Euclidean spanners. *Int. J. Comp. Geom. and Applic.*, 7(4):297–315, 1997.

8 Mohammad Farshi and Joachim Gudmundsson. Experimental study of geometric t-spanners. *ACM Journal of Experimental Algorithmics*, 14, 2009. URL: `http://doi.acm.org/10.1145/1498698.1564499`, `doi:10.1145/1498698.1564499`.

Lions and contamination, triangular grids, and Cheeger constants

Henry Adams, Leah Gibson, and Jack Pfaffinger

Abstract

Suppose each vertex of a graph is originally occupied by contamination, except for those vertices occupied by lions. As the lions wander on the graph, they clear the contamination from each vertex they visit. However, the contamination simultaneously spreads to any adjacent vertex not occupied by a lion. How many lions are required in order to clear the graph of contamination? We give a lower bound on the number of lions needed in terms of the Cheeger constant of the graph. Furthermore, the lion and contamination problem has been studied in detail on square grid graphs by Brass et al. and Berger et al., and we extend this analysis to the setting of triangular grid graphs.

Related Version arXiv:2012.06702v2

1 Introduction

In the "lions and contamination" pursuit-evasion problem [1, 2, 5], lions are tasked with clearing a square grid graph consisting of vertices and edges. At the start of the problem, all vertices occupied by lions are considered cleared of contamination, and the rest of the vertices are contaminated. The lions move along the edges of the grid, and each new vertex they occupy becomes cleared. However, the contamination can also travel along the edges of the grid not blocked by a lion and re-contaminate previously cleared vertices. How many lions are needed to clear the grid?

Certainly n lions can clear an $n \times n$ grid graph by sweeping from one side to the other. One might conjecture that n lions are required to clear an $n \times n$ grid graph. However, in general it is not yet known whether $n-1$ lions suffice or not. As a lower bound, the paper [2] proves that at least $\lfloor \frac{n}{2} \rfloor + 1$ lions are required to clear an $n \times n$ grid graph. The details of the discretization certainly matter, in the following sense. For a $n \times n \times n$ grid graph, one might expect that n^2 lions are required, but [1] shows that when $n = 3$, only $8 = n^2 - 1$ lions suffice to clear a $3 \times 3 \times 3$ grid.

We consider the case of planar triangular grid graphs, under various models of lion motion. Given $R_{n,l}$, a planar parallelogram of height n vertices and length l vertices triangulated by equilateral triangles, n lions suffice to clear $R_{n,l}$. However, we conjecture that n lions do not suffice when all lions must move simultaneously. In the setting of simultaneous motion (which we refer to as "caffeinated lions," as the lions never take a break), we show that $\lfloor \frac{3n}{2} \rfloor$ lions suffice to clear $R_{n,l}$. Furthermore, via a comparison with [1, 2], we show that $\lfloor \frac{n}{2} \rfloor$ lions are insufficient to clear a triangulated rhombus, in which each side of the rhombus has length n. Lastly, for an equilateral triangle discretized into smaller triangles, with n vertices per side, we conjecture that $\frac{n}{2\sqrt{2}}$ lions are not sufficient to clear the graph.

In the setting of an arbitrary graph G, we give a lower bound on the number of lions needed to clear the graph in terms of the *Cheeger constant* of the graph. The Cheeger constant, roughly speaking, is a measure of how hard it is to disconnect the graph into two pieces of approximately equal size by cutting edges [4]. The use of the Cheeger constant in graph theory is inspired by its successful applications in Riemannian geometry [3]. Our bound on the number of lions in terms of a graph's Cheeger constant is quite general (it holds for any graph), and therefore we do not expect it to be sharp for any particular graph. We use Cheeger constants for "polite lions" where only one lion may move at a time. If we

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

let G be a connected graph with vertex set V and with volume Cheeger constant g, then if $k \leq \frac{1}{2}\lfloor\frac{|V|}{2}\rfloor g$, then G cannot be cleared by k polite lions. The advantage of using the volume Cheeger constant is that it gives a lower bound on the number of lions needed to clear an arbitrary graph. That is not to say that the bound obtained by this method is near the optimal number of lions. Rather, Theorems 2 and 3 gives a weak bound for any graph, including graphs that do not have obvious symmetry that can be used to discover a better bound.

We explain the connection to the Cheeger constant in Section 2, and give one result on triangular grid graphs in Section 3.

2 Connection to Cheeger constant

In graph theory, the term *Cheeger constant* refers to a numerical measure of how much of a bottleneck a graph has. The term arises from a related quantity, also known as a Cheeger constant, that is used in differential geometry: the Cheeger constant of a Riemannian manifold depends on the minimal area of a hypersurface that is required to divide the manifold into two pieces [3]. For both graphs and manifolds, the Cheeger constant can be used to provide lower bounds on the eigenvalues of the Laplacian (of the graph or of the manifold). In the graph theory literature, there are several different quantities known as the Cheeger constant, and they are each defined slightly differently. For some more background and applications of Cheeger constants to graphs, see [4, Chapter 2]. In this section, we show a relationship between the Cheeger constant of a graph and the number of lions needed to clear this graph of contamination.

Let us simplify the lion and contamination problem a bit. Recall a graph with *polite lions* is one in which at each turn, at most one lion can move. We will now define a new value for our graph, which we will call the *volume Cheeger constant*.

▶ **Definition 1.** For a graph G, let the S be a subset of vertices and define the volume Cheeger constant to be

$$g := \min\left\{\frac{|\partial S|}{\min\{|S|, |\overline{S}|\}} : S \subseteq V(G),\ S \neq \emptyset,\ S \neq V(G)\right\},$$

where $\partial S = \{v \in S \mid uv \in E(G),\ u \notin S\}$ (i.e. the set of boundary vertices), and $\overline{S}$ is the set $V(G) \setminus S$. Note that ∂S is defined differently than the vertex boundary used in [4] (for us, ∂S is a subset of S instead of $\overline{S}$), but this doesn't affect the value of the volume Cheeger constant).

For a connected graph, we have $0 < g \leq 1$. For the upper bound, consider a connected graph G. If $|S| = 1$, then $\min\{|S|, |\overline{S}|\} = 1$ and since G is connected, $|\partial S| = 1$. Thus any connected graph has a volume Cheeger constant at most 1 since g takes the minimum of all vertex subsets. For the lower bound, note that if S is neither empty nor all of $V(G)$, then $|\partial S| \geq 1$ when G is connected. If G is disconnected then $g = 0$ since $|\partial S| = 0$ if you take S to be a connected component. Roughly speaking, a larger g tells us that overall the graph has more connections; a smaller g says that the graph is easier to break into pieces. Consider the graphs in Figure 1 for a few examples of Cheeger constants.

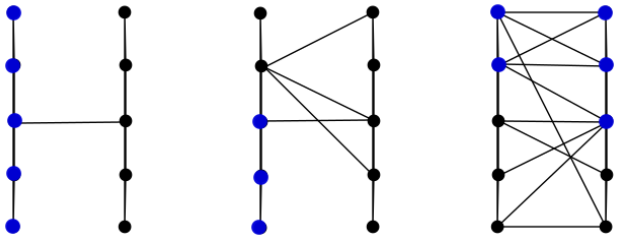


Figure 1 From left to right, the volume Cheeger constants are $g = \frac{1}{5}, g = \frac{1}{3}, g = \frac{2}{5}$. These Cheeger constants can be realized by taking the blue vertices to be set S.

We now give lower bounds on the number of lions needed to clear a graph in terms of the volume Cheeger constant. We do this first for polite lions, and then next for arbitrary lions.

▶ **Theorem 2.** *Let G be a connected graph with vertex set V and with volume Cheeger constant g. If $k \leq \frac{1}{2}\lfloor\frac{|V|}{2}\rfloor g$, then G cannot be cleared by k polite lions.*

Proof. Suppose that $k \leq \frac{1}{2}\lfloor\frac{|V|}{2}\rfloor g$ impolite lions can clear G. Since the number of cleared vertices can increase by at most one in each step with polite lions, in the process of clearing there must be a time t satisfying $|C(t)| = \lfloor\frac{|V|}{2}\rfloor$. Now, by the definition of the Cheeger constant g, we have $|\partial C(t)| \geq \lfloor\frac{|V|}{2}\rfloor g$. Since $2k \leq |\partial C(t)|$, this implies by Lemma 5 from the following section that $|C(t+1)| \leq |C(t)|$. Therefore the number of cleared vertices is always at most $\lfloor\frac{|V|}{2}\rfloor$, contradicting the clearing of G with k lions. ◀

▶ **Theorem 3.** *Let G be a connected graph with volume Cheeger constant g. If $k \leq \frac{g|V|}{4+g}$, then G cannot be cleared by k lions.*

Proof. If $k \leq \frac{g|V|}{4+g}$, then it follows that $k(2+\frac{g}{2}) \leq \frac{g|V|}{2}$, and hence $2k \leq g(\frac{|V|}{2} - \frac{k}{2})$. This implies that for any x satisfying $0 \leq x \leq \frac{k}{2}$, we have $2k \leq g(\frac{|V|}{2} - x)$. By definition of the volume Cheeger constant, $g \leq \frac{|\partial S|}{\frac{|V|}{2}-x}$, where S is any subset of V of size $\frac{|V|}{2} \pm x$. Combining these two facts implies that for any x satisfying $0 \leq x \leq \frac{k}{2}$, we have $2k \leq g(\frac{|V|}{2} - x) \leq |\partial S|$, where $S \subseteq V$ is any set of size $|S| = \frac{|V|}{2} \pm x$. By Lemma 4 from the following section the size of the cleared set can increase by at most k in any step. Therefore there is some time t when the number of cleared vertices is within $\pm\frac{k}{2}$ of $\frac{|V|}{2}$ and when $|C(t+1)| > |C(t)|$. But since $2k \leq |\partial C(t)|$ at this time step t, Lemma 5 then implies that the number of cleared vertices cannot increase in the next set, giving a contradiction. Therefore k lions do not suffice to clear graph G. ◀

The advantage of using the volume Cheeger constant is that it gives a lower bound on the number of lions needed to clear an arbitrary graph. That is not to say that the bound obtained by this method is near the optimal number of lions. Rather, Theorems 2 and 3 give a weak bound for any graph, including graphs that do not have obvious symmetry that can be used to discover a better bound.

For example, if the volume Cheeger constant is $g = 1$, which happens if G is a complete graph, then Theorem 3 implies that at least $|V|/5$ lions are needed to clear the graph G. If $g = \frac{1}{2}$, then Theorem 3 says that at least $|V|/9$ lions are needed.

3 Insufficiency of $\lfloor \frac{n}{2} \rfloor$ lions on a triangulated square

We will use some of the methods and proofs from [1] to show that $\lfloor \frac{n}{2} \rfloor$ (non-caffeinated) lions cannot clear $R_n := R_{n,n}$ where $\mathbb{R}_{n,n}$ is a planar graph which forms a parallelogram of height n vertices and length n vertices, subdivided into a grid of equilateral. In this proof, we stretch the "rhombus" graph R_n to instead be drawn as a square triangulated by right triangles. It should be noted that this grid holds all of the same properties as before (the isomorphism type of the graph is unchanged).

We define S_n to be the $n \times n$ square grid graph discussed in [1]; see Figure 2. When each square is subdivided via a diagonal edge, drawn from the top left to the bottom right, then we obtain a graph isomorphic to R_n as shown in Figure 3.

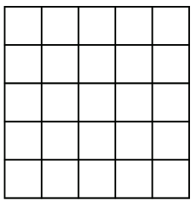


Figure 2 The graph S_5.

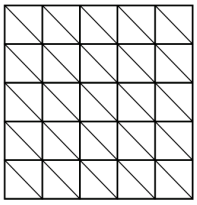


Figure 3 R_5 with right triangles.

The following two lemmas from [1] hold in an arbitrary graph.

▶ **Lemma 4** (Lemma 1 of [1]). *Let k be the number of lions on a graph. The number of cleared vertices cannot increase by more than k in one time step.*

▶ **Lemma 5** (Lemma 2 of [1]). *Let k be the number of lions. If there are at least $2k$ boundary vertices in the set $C(t)$ of cleared vertices, then the number of cleared vertices cannot increase in the following step: $|\partial C(t)| \geq 2k$ implies $|C(t+1)| \leq |C(t)|$.*

For the specific case of square grid graphs S_n, [1] defines a "fall-down transformation". This transformation T takes any subset of the vertices of S_n, and maps it to a (potentially different) subset of the same size. The first step in the fall-down transformation, roughly speaking, is to allow gravity to act on the subset of vertices, so that each vertex falls as far as possible towards the bottom of its column. The number of vertices in any column remains unchanged by this step. The second step in the fall-down transformation is to then allow a horizontal force to act on the current subset of vertices, so that each vertex moves as far as possible to the left-hand side of its row, maintaining the number of vertices in any row.

In the case of a square grid S_n, [1] proves that the fall-down transformation does not increase the number of boundary vertices in a set:

▶ **Lemma 6** (Lemma 4 of [1]). *In the graph S_n, the fall-down transformation T is monotone, meaning that the number of boundary vertices in a subset S of vertices from S_n does not increase upon applying the fall-down transformation.*

Since the vertex set of S_n is the same as the vertex set of R_n, we immediately get a fall-down transformation T that maps a subset of vertices in R_n to a subset of vertices in R_n. We will show that this new fall-down transformation has the same monotonicity property, which is not a priori clear as the boundary of a set of vertices in S_n might be smaller than the boundary of that same set of vertices in R_n. Another comment is that when defining

the fall down transformation T on S_n, the choice of down vs. up or of left vs. right does not matter. But these choices do matter when defining a fall-down transformation on R_n, due to the diagonal edges as drawn in Figures 3 and 4. We want to emphasize we have chosen to map down and to the left; the following lemma in part depends on this choice.

▶ **Lemma 7.** *Let S be a set of vertices in R_n (or equivalently, in S_n). The set of boundary vertices of $T(S)$ in R_n is the same as the set of boundary vertices of $T(S)$ in S_n.*

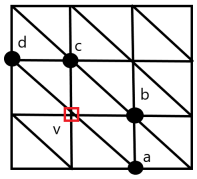


Figure 4 R_n.

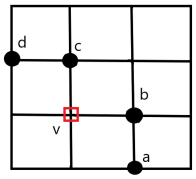


Figure 5 S_n.

Proof. We will show that the set of boundary vertices of $T(S)$ in R_n is the same as the set of boundary vertices of $T(S)$ in S_n. First suppose that a vertex v of $T(S)$ is a boundary vertex in R_n. Referring to the diagram in Figure 4, this implies that one of d, c, b or a is not in $T(S)$. However, we can reduce this to the case that one of c or b is not in $T(S)$, since $d \notin T(S) \Rightarrow c \notin T(S)$, and since $a \notin T(S) \Rightarrow b \notin T(S)$. If $c \notin T(S)$, then v is a boundary vertex of $T(S)$ in both R_n and S_n. The same is true if $b \notin T(S)$. This proves that if v is a boundary vertex of $T(S)$ in R_n, then it is a boundary vertex of $T(S)$ in S_n. But, referring to Figure 5, if v is a boundary vertex of $T(S)$ in S_n, then one of c or b is not in $T(S)$, which implies that v is a boundary vertex of $T(S)$ in R_n as well. Therefore the set of boundary vertices of $T(S)$ in R_n is the same as the set of boundary vertices of $T(S)$ in S_n. ◀

We know that T is monotone on S_n, i.e. that the number of boundary vertices does not increase as a result of applying T. Lemma 7 therefore immediately implies that T is also monotone on R_n, i.e. that the number of boundary vertices of a set S in R_n is no more than the number of boundary vertices of $T(S)$ in R_n.

▶ **Corollary 8.** *In the graph R_n, the fall-down transformation T is monotone, meaning that the number of boundary vertices in a subset S of vertices from R_n does not increase upon applying the fall-down transformation.*

This now brings us to the last lemma.

▶ **Lemma 9** (Lemma 5 of [1])**.** *Any vertex set S on R_n satisfying $\frac{n^2}{2} - \frac{n}{2} < |S| < \frac{n^2}{2} + \frac{n}{2}$ has at least n boundary vertices.*

Proof of Lemma 9. By Lemma 6, we know that the fall-down transformation T acts monotonically on the number of boundary vertices in R_n. Thus the proof of Lemma 5 from [1] will also hold for our grid graph with diagonal edges added. ◀

▶ **Theorem 10.** *$\lfloor \frac{n}{2} \rfloor$ lions do not suffice to clear R_n.*

Proof. Let the number of lions be $k \leq \lfloor \frac{n}{2} \rfloor$. The lions will eventually have to clear all n^2 vertices. By Lemma 5, we know that $|C(t+1)| - |C(t)| \leq k \leq \frac{n}{2}$ for all times t. Thus there must be a time t where $\frac{n^2}{2} - \frac{n}{4} \leq |C(t)| \leq \frac{n^2}{2} + \frac{n}{4}$ and $|C(t+1) \geq |C(t)|$. But by Lemma 9, there are at least n boundary vertices of $C(t)$ at time t, and so Lemma 5 tells us that $|C(t+1)| \leq |C(t)|$, which is a contradiction. Thus $k \leq \lfloor \frac{n}{2} \rfloor$ lions do not suffice to clear R_n. ◀

References

1 Florian Berger, Alexander Gilbers, Ansgar Grüne, and Rolf Klein. How many lions are needed to clear a grid? *Algorithms*, 2(3):1069–1086, Sep 2009.

2 Peter Brass, Kyue D Kim, Hyeon-Suk Na, and Chan-Su Shin. Escaping off-line searchers and a discrete isoperimetric theorem. In *International Symposium on Algorithms and Computation*, pages 65–74. Springer, 2007.

3 Jeff Cheeger. A lower bound for the smallest eigenvalue of the Laplacian. In *Proceedings of the Princeton conference in honor of Professor S. Bochner*, pages 195–199, 1969.

4 Fan R. K. Chung. *Spectral graph theory*. Regional conference series in mathematics. Published for the Conference Board of the mathematical sciences by the American Mathematical Society, 1997.

5 Adrian Dumitrescu, Ichiro Suzuki, and Paweł Żyliński. Offline variants of the "lion and man" problem: Some problems and techniques for measuring crowdedness and for safe path planning. *Theoretical Computer Science*, 399(3):220–235, 2008.

A Geometric Approach to Inelastic Collapse*

Bernard Chazelle[1], Kritkorn Karntikoon[2], and Yufei Zheng[3]

1 **Department of Computer Science, Princeton University**
`chazelle@cs.princeton.edu`
2 **Department of Computer Science, Princeton University**
`kritkorn@cs.princeton.edu`
3 **Department of Computer Science, Princeton University**
`yufei@cs.princeton.edu`

Abstract

We show in this note how to interpret logarithmic spiral tilings as one-dimensional particle systems undergoing inelastic collapse. By deforming the spirals appropriately, we can simulate collisions among particles with distinct or varying coefficients of restitution. Our geometric constructions provide a strikingly simple illustration of a widely studied phenomenon in the physics of dissipative gases: the collapse of inelastic particles.

1 Introduction

Collisions in a granular gas preserve momentum but not kinetic energy. Interactions are dissipative, with the velocities of two colliding particles governed by a stochastic matrix $\left(\begin{smallmatrix} p & q \\ q & p \end{smallmatrix}\right)$, for $p \leq 1/2$. When the coefficient of restitution, defined as $r = 1 - 2p$, is less than 1, the collisions are inelastic and the particles may collapse to a single point in a finite amount of time: this intriguing phenomenon of *inelastic collapse* was first investigated in one dimension by Bernu & Mazighi [2] and McNamara & Young [6]. Further studies and extensions to a larger number n of particles were given in [1, 2, 3, 4, 5, 6, 7, 8]. In the case $n = 3$, inelastic collapse requires $r < 7 - 4\sqrt{3}$ [4, 6, 7], while in general the requirement is that $n \gtrsim 2(\ln 2)/(1 - r)$. Matching constructions for large n exist but entail intricate eigenvalue estimates [1, 2]. We rederive these bounds by simple geometric means, and we also extend them to other types of collisions. Our particle systems are derived from one-dimensional projections of spiral tilings of a disk (see §2). Using different spirals allows the presence of particles with different coefficients of restitution (see §3). The notable feature of our arguments is to be entirely geometric.

2 The Inelastic Collapse of Identical Particles

We describe the dynamics of n identical particles moving towards the center of a disk and colliding along the way. The one-dimensional system is derived by projection to a line. We begin with the geometry of the system, which is a quadrilaterial tiling of the complex unit disk by logarithmic spirals.

2.1 Spiral tilings

Fix $0 < \lambda_o < 1$ and let $\mathcal{C}_\alpha = \left\{ \lambda_o^{|\varphi - \alpha|} e^{i\varphi} \,|\, \varphi \in \mathbb{R} \right\}$. The curve $\mathcal{C}_\alpha$ consists of two logarithmic spirals running clockwise and counterclockwise from the point $e^{i\alpha}$. The family $\{\mathcal{C}_\alpha\}_{0 \leq \alpha < 2\pi}$

* This work was supported in part by NSF grant CCF-2006125.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

forms two foliations of the unit complex disk $\mathcal{D}$ (minus the origin). Whereas no pair of spirals going in the same direction meet, the other pairs intersect infinitely often along the diameter bisecting their starting points. Fix an integer $n > 2$ and write $\theta = \pi/n$. We rectify the spiral $\mathcal{C}_\alpha$ by creating the vertices $\lambda_o^{|k\theta-\alpha|}e^{ik\theta}$ for all $k \in \mathbb{Z}$; then we join consecutive pairs by straightline segments, which produces the polygonal spiral $\mathcal{C}_\alpha^R$ in Figure 1(i).

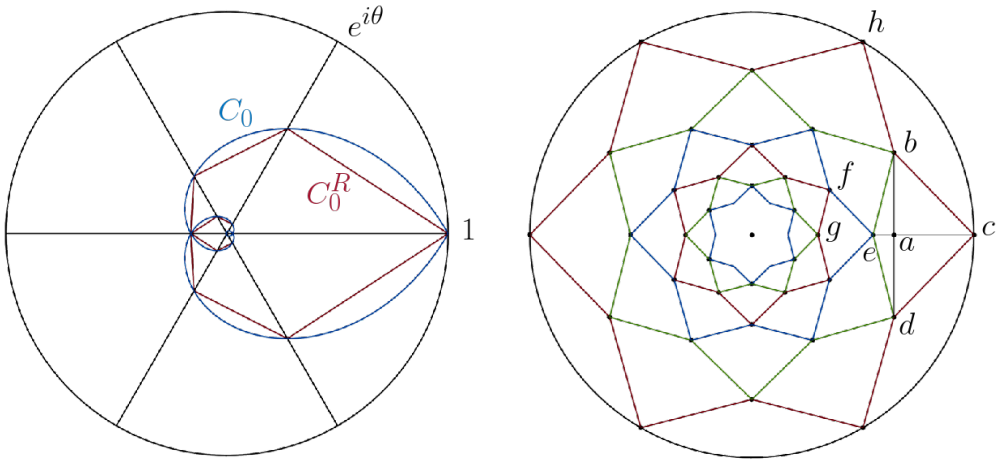


Figure 1 (i) The spirals $\mathcal{C}_\alpha$ and $\mathcal{C}_\alpha^R$, for $\alpha = 0$ and $\theta = \pi/3$; (ii) an (n, λ)-tiling for a system of $2n = 12$ colliding particles.

The collection of polygonal curves $\left\{ \mathcal{C}_{2j\theta}^R \,|\, 0 \le j < n \right\}$ forms an infinite sequence of nested concentric similar $2n$-gons $P_k := \lambda e^{i\theta} P_{k-1}$, where $\lambda = \lambda_o^\theta$ and P_0 is the outer "star" shown in Figure 1(ii): its vertices $e^{il\theta}\lambda^{(1-(-1)^l)/2}$ run in counterclockwise order ($0 \le l < 2n$). To ensure that the shape is indeed a star, every other vertex of P_0 needs to be reflex, which requires that $\lambda < \cos\theta$. This partitions the polygon P_0 into an infinite collection of similar convex quadrilaterals, which forms an (n, λ)-*tiling*. We define the *fundamental ratio* $\rho := ae/ac$ of the (n, λ)-tiling and justify its name by noting that it is independent of the polygon P_k used to define it. Referring to Figure 1(ii), we observe that $ac = 1 - \lambda\cos\theta$ and $ae = \lambda\cos\theta - \lambda^2$ and that, for any $0 < \lambda < \cos\theta$,

$$\rho = \frac{\lambda(\cos\theta - \lambda)}{1 - \lambda\cos\theta} \qquad \text{and} \qquad 0 < \rho < 1\,. \tag{1}$$

2.2 Particles traveling in a disk

Place two particles at each one of the n outer vertices of P_0 and set them in motion along the two incident edges with a speed equal to bc. We show below that the particles will zigzag toward the center (as in the trajectory $c, b, e, f, g, \ldots$) provided that the coefficient of restitution r is equal to $\rho < 1$, where $r = 1 - 2p$; recall that, whenever two particles with velocities $u, v \in \mathbb{C}$ collide, they bounce away from each other and update their velocities as follows:

$$\begin{pmatrix} u \\ v \end{pmatrix} \leftarrow \begin{pmatrix} p & q \\ q & p \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix};$$

where $0 < p < q < 1$ and $p + q = 1$.

▶ **Lemma 2.1.** *The $2n$ particles travel along the edges of the tiling through pairwise collisions if and only if the fundamental ratio ρ is equal to the coefficient of restitution r. If each particle spends one unit of time on the boundary ∂P_0, then it travels on ∂P_k for a duration of δ^k, where $\delta = \lambda^2/\rho$. The total travel time is bounded if and only if $\lambda < \frac{1}{\cos\theta} - \tan\theta$, in which case it is equal to $1/(1-\delta)$.*

Proof. For convenience, we tilt the tiling by θ to put b and f on the X-axis (Figure 2). Two particles travel from c and h to b with velocity u and v respectively. The first one bounces at b and proceeds with velocity $u' = pu + qv$. Since $u_x = v_x$ and $u_y = -v_y$, we have $u'_x = u_x$ and $u'_y = -ru_y$; therefore $|slope(u')| = r|slope(u)|$. By similarity, bc and ef are parallel; hence $|slope(u')| = r|slope(ef)|$. The consistency of the particle collision with the tiling means that u' should be parallel to the segment be. The condition thus becomes $|slope(be)| = r|slope(ef)|$; hence $r = mf/mb = \rho$.

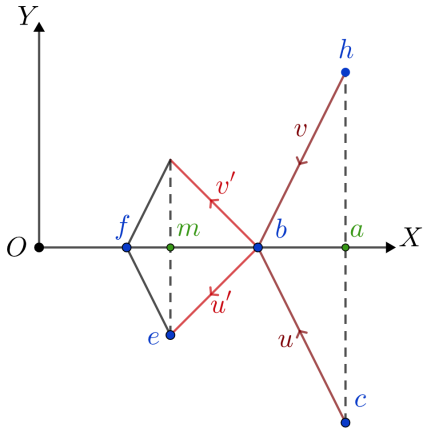


Figure 2 How colliding particles follow the edges of the (n, λ)-tiling. The coefficient of restitution must be equal to the ratio $\rho = mf/mb$.

If the particle travels from c to b in one unit of time, then $u_y = ac$ and $u'_y = -ru_y = -rac$. It follows that the time δ for the particle to bounce from b to e is equal to $me/|u'_y| = \frac{1}{r}me/ac = \lambda^2/r$. More generally, δ is the ratio between the time spent on be and that spent on cb. By symmetry, the same ratio δ holds between the travel times along any two consecutive edges on the trajectory. This follows from the fact that the travel time along an edge is itself a ratio length/speed and that, from one boundary ∂P_k to the next, ∂P_{k+1}, the ratio between consecutive lengths is independent of k and the same is true of consecutive speeds. This implies a travel time of δ^k on ∂P_k. Convergence implies that $\delta < 1$, which, by (1), means that λ must be less than the smaller root of $\lambda^2 \cos\theta - 2\lambda + \cos\theta$ (since the larger one exceeds 1). This gives us the inequality $\lambda < (1 - \sin\theta)/\cos\theta$. Note that this condition is not implied by the previous requirement that $0 < \lambda < \cos\theta$. ◀

By (1), setting $r = \rho$ for any $\lambda < \cos\theta$ produces a valid particle system traveling inward through the (n, λ)-tiling. Of course, the interesting question is whether this holds for *any* value of the coefficient of restitution. We address this issue below in the context of one-dimensional systems.

2.3 One-dimensional collapse

The real parts of the $2n$ particles' positions in the unit disk $\mathcal{D}$ describe a one-dimensional particle system. To see why, it is useful to distinguish between the *positive* particles, those numbered $1, \ldots, n$ counterclockwise around $\mathcal{D}$, from the others, the *negative* particles. The name comes from the fact that the positive (resp. negative) particles always remain in the upper (resp. lower) complex halfplane. Each positive particle j is naturally paired with the negative particle $2n + 1 - j$, since their trajectories are conjugate. Particles can only collide with other particles of the same sign or with their conjugates; in the latter case, the collision does not alter the motion along the real axis. All the other collisions occur in conjugate pairs. This shows that the real-axis motion of the positive particles alone constitutes a bona fide collision system over n particles with the same coefficient of restitution.

▶ **Theorem 2.2.** *Fix any integer $n > 2$, and write $\theta = \pi/n$ and $r_0 = (1 - \sin\theta)/(1 + \sin\theta)$. Given any positive coefficient of restitution $r \le r_0$, there is a scaling factor λ such that the line projection of the (n, λ)-tiling forms the trajectory of a one-dimensional n-particle system exhibiting inelastic collapse. The collapse time is $r/\big(r - \lambda^2\big)$ for any $r < r_0$ and $\lambda = q\cos\theta - (q^2\cos^2\theta - r)^{1/2}$, where $q = (1 + r)/2$.*

Proof. Setting $r = \rho$ in (1) yields the quadratic equation

$$\lambda^2 - 2q(\cos\theta)\lambda + r = 0; \tag{2}$$

hence $\lambda = q\cos\theta \pm \sqrt{q^2\cos^2\theta - r}$. The roots need to be real; hence $\sin\theta \le p/q$ or, equivalently, $r \le r_0$. We verify that $0 < \lambda < \cos\theta$, as required of a valid (n, λ)-tiling, which is a consequence of $\sqrt{q^2\cos^2\theta - r} < p\cos\theta$. By Lemma 2.1, the collapse time is infinite if $\delta = \lambda^2/r \ge 1$ and equal to $\sum_{k\ge0}\delta^k = 1/(1-\delta) = r/(r-\lambda^2)$ if $\delta < 1$. The smaller root of (2), if strictly smaller, always satisfies the latter condition while the larger one never does. This follows from the fact that $\lambda_-\lambda_+ = r$, $q\cos\theta \ge \sqrt{r}$, and $\lambda_+ \ge q\cos\theta$; hence $\lambda_+^2 \ge r$. ◀

In our construction, the upper bound on the coefficient of restitution is $(1-\sin\theta)/(1+\sin\theta)$. As n goes to infinity, this gives us $n \gtrsim 2\pi/(1-r)$, which matches the bounds from [1, 2]. For $n = 3$, our construction rediscovers the classic bound of $7 - 4\sqrt{3}$ [4, 6, 7].

3 Distinct Coefficients of Restitution

Our construction does not require a fixed scaling λ. Instead of placing the vertices on circles of radius λ^k for $k \ge 0$, we can use an arbitrary decreasing radius sequence $(\lambda_k)_{k\ge0}$, with $\lambda_0 = 1$. We assign a coefficient restitution r_k for the collisions at radius λ_k; the dependency on k might reflect a gain or loss of elasticity after repeated collisions. For notational convenience, let $p = (1 - r_1)/2$, $\lambda = \lambda_1$, and $\mu = \lambda_2$. By reference to Figure 3, we now kick a particle from a to b with velocity $u = b - a$ (using complex numbers), and one from c to b with velocity $v = b - c$. Post-collision, the first particle travels from b to d with velocity $u' = pu + (1 - p)v = \sigma_1(d - b)$, for some $\sigma_1 > 0$; hence $b - c + p(c - a) = \sigma_1(d - b)$. Since $a = 1$, $b = \lambda e^{i\theta}$, $c = e^{2i\theta}$, and $d = \mu$, we divide the equation by $e^{i\theta}$ and find that

$$\lambda - e^{i\theta} + 2pi\sin\theta = \sigma_1\big(\mu e^{-i\theta} - \lambda\big);$$

therefore, $\lambda - \cos\theta = \sigma_1(\mu\cos\theta - \lambda)$ and $r_1 = \sigma_1\mu$. More generally, for $k > 0$, we replace λ and μ by λ_k and λ_{k+1}, respectively, and we scale the relations by λ_{k-1}:

$$\sigma_k = \frac{\lambda_{k-1}\cos\theta - \lambda_k}{\lambda_k - \lambda_{k+1}\cos\theta} \qquad \text{and} \qquad r_k = \frac{\cos\theta - \lambda_k/\lambda_{k-1}}{\lambda_k/\lambda_{k+1} - \cos\theta}\,. \tag{3}$$

Of course, we retrieve the relation $r = \rho$ in (1) in the case $\lambda_k = \lambda^k$ corresponding to having fixed coefficients of restitution.

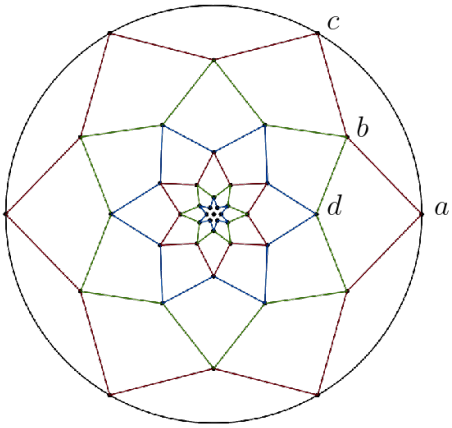


Figure 3 An irregular tiling.

3.1 Finite-time inelastic collapse

From the relation $u' = \sigma_1(d - b)$, we see that the time spent crossing bd is precisely $1/\sigma_1$. More generally, $1/\sigma_k$ is the time spent on the $(k+1)$-st star polygon, given a unit travel time on the previous polygon. It follows that the total travel duration is the sum of all the products of the form $1/\sigma_1 \cdots \sigma_k$, which is

$$1 + \sum_{k=1}^{\infty} \prod_{j=1}^{k} \frac{\lambda_j - \lambda_{j+1} \cos\theta}{\lambda_{j-1} \cos\theta - \lambda_j}. \tag{4}$$

By projection onto the real line, finite-time inelastic collapse is guaranteed if

$$\lambda_{k+1} \geq \frac{1+c}{\cos\theta} \lambda_k - c\lambda_{k-1},$$

for some fixed $c < 1$. Again, we can check that, if $\lambda_k = \lambda^k$, then bounded travel time means that $\lambda < \frac{1}{\cos\theta} - \tan\theta$, as claimed in Lemma 2.1.

3.2 Red-blue particles

Consider two species of particles, blue and red. The blue particles collide together with the coefficient of restitution r_1 and the same is true of the red ones. Particles of different colors, however, collide with the coefficient r_2. Arrange the particles as usual, with the sequence blue, blue, red, red, blue, blue, red, red, etc. Set the scaling factor $\lambda_k = \mu^j$ if $k = 2j$, and $\lambda_k = \lambda\mu^j$ if $k = 2j+1$. By (3), we choose

$$r_1 = \frac{\mu(\cos\theta - \lambda)}{\lambda - \mu\cos\theta} \qquad \text{and} \qquad r_2 = \frac{\lambda\cos\theta - \mu}{1 - \lambda\cos\theta}.$$

Each factor in (4) is of the form

$$\frac{\lambda_j - \lambda_{j+1}\cos\theta}{\lambda_{j-1}\cos\theta - \lambda_j} = \begin{cases} \mu(1 - \lambda\cos\theta)/(\lambda\cos\theta - \mu) = \mu/r_2 & \text{if } j \text{ is even} \\ (\lambda - \mu\cos\theta)/(\cos\theta - \lambda) = \mu/r_1 & \text{else}. \end{cases}$$

The travel time is finite if $\mu^2 < r_1 r_2$, which is

$$\mu(\lambda - \mu\cos\theta)(1 - \lambda\cos\theta) < (\cos\theta - \lambda)(\lambda\cos\theta - \mu).$$

References

1 D. Benedetto and E. Caglioti. The collapse phenomenon in one-dimensional inelastic point particle systems. *Physica D: Nonlinear Phenomena*, 132(4):457 – 475, 1999.

2 B. Bernu and R. Mazighi. One-dimensional bounce of inelastically colliding marbles on a wall. *Journal of Physics A: Mathematical and General*, 23(24):5745 – 5754, 1990.

3 B. Cipra, P. Dini, S. Kennedy, and A. Kolan. Stability of one-dimensional inelastic collision sequences of four balls. *Physica D: Nonlinear Phenomena*, 125(3):183 – 200, 1999.

4 P. Constantin, E. Grossman, and M. Mungan. Inelastic collisions of three particles on a line as a two-dimensional billiard. *Physica D: Nonlinear Phenomena*, 83(4):409 – 420, 1995.

5 S. McNamara. Inelastic collapse. pages 267 – 277, 2002.

6 W.R. Young. S. McNamara. Inelastic collapse and clumping in a one-dimensional granular medium. *Physics of Fluids A: Fluid Dynamics*, 4(3):496 – 504, 1992.

7 K. Shida and T. Kawai. Cluster formation by inelastically colliding particles in one-dimensional space. *Physica A: Statistical Mechanics and its Applications*, 162(1):145 – 160, 1989.

8 L.P. Kadanoff. T. Zhou. Inelastic collapse of three particles. *Physical review E*, 54(1):623 – 628, 1996.

Max-Min 3-dispersion on a Convex Polygon

Yasuaki Kobayashi[1], Shin-ichi Nakano[2], Kei Uchizawa[3], Takeaki Uno[4], Yutaro Yamaguchi[5], and Katsuhisa Yamanaka[6]

1 **Kyoto University, Japan**
kobayashi@iip.ist.i.kyoto-u.ac.jp
2 **Gunma University, Japan**
nakano@cs.gunma-u.ac.jp
3 **Yamagata University, Japan**
uchizawa@yz.yamagata-u.ac.jp
4 **National Institute of Informatics, Japan**
uno@nii.jp
5 **Kyushu University, Japan**
yutaro_yamaguchi@inf.kyushu-u.ac.jp
6 **Iwate University, Japan**
yamanaka@cis.iwate-u.ac.jp

Abstract

Given a set P of n points and an integer k, we wish to place k facilities on points in P so that the minimum distance between facilities is maximized. The problem is called the k-dispersion problem, and the set of such k points is called a k-dispersion of P. Note that the 2-dispersion problem corresponds to the computation of the diameter of P. Thus the k-dispersion problem is a natural generalization of the diameter problem. In this paper we consider the case of $k = 3$, which is the 3-dispersion problem, when P is in convex position. We give an $O(n^2)$-time algorithm to compute a 3-dispersion of P.

1 Introduction

The facility location problem and many of its variants have been studied [11, 12]. Typically, given a set P of points in the Euclidean plane and an integer k, we wish to place k facilities on points in P so that a designated function on distance is minimized. In contrast, in the *dispersion problem*, we wish to place facilities so that a designated function on distance is maximized.

The intuition of the problem is as follows. Assume that we are planning to open several coffee shops in a city. We wish to locate the shops mutually far away from each other to avoid self-competition. So we wish to find k points so that the minimum distance between the shops is "maximized". See more applications, including *result diversification*, in [9, 20, 21].

Now, we define the *max-min k-dispersion problem*. Given a set P of n points in the Euclidean plane and an integer k with $k < n$, we wish to find a subset $S \subset P$ with $|S| = k$ in which the cost $\mathrm{cost}(S) = \min_{u,v \in S} d(u, v)$ is maximized, where $d(u, v)$ is the distance between u and v in P. Such a set S is called a k-dispersion of P. This is the max-min version of the k-dispersion problem [20, 24]. Several heuristics to solve the problem are compared [14]. The max-sum version [6, 7, 8, 9, 10, 15, 17, 20] and a variety of related problems [4, 6, 10] are studied.

The max-min k-dispersion problem is NP-hard even when the triangle inequality is satisfied [13, 24]. An exponential-time exact algorithm for the problem is known [2]. The running time is $O(n^{\omega k/3} \log n)$, where $\omega < 2.373$ is the matrix multiplication exponent.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 79, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

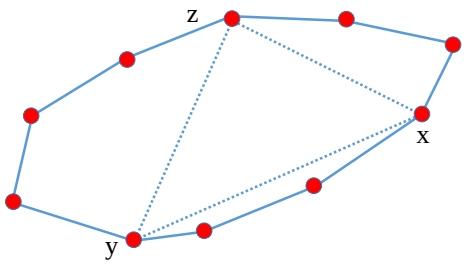


Figure 1 An example of 3-dispersion. $\{x, y, z\}$ is a 3-dispersion.

The problem in the D-dimensional Euclidean space can be solved in $O(kn)$ time for $D = 1$ if a set P of points are given in the order on the line and is NP-hard for $D = 2$ [24]. One can also solve the case $D = 1$ in $O(n \log \log n)$ time [3] by the sorted matrix search method [16] (see a good survey for the sorted matrix search method in [1, Section 3.3]), and in $O(n)$ time [2] by a reduction to the path partitioning problem [16]. Even if a set P of points are not given in the order on the line the running time for $D = 1$ is $O((2k^2)^k n)$ [5]. Thus if k is a constant we can solve the problem in $O(n)$ time.

If P is a set of points on a circle, and the points in P are given in the order on the circle, and the distance between them is the distance along the circle, then one can solve the k-dispersion problem in $O(n)$ time [23].

For approximation, the following results are known. Ravi et al. [20] proved that, unless P = NP, the max-min k-dispersion problem cannot be approximated within any constant factor in polynomial time, and cannot be approximated with a factor less than two in polynomial time when the distance satisfies the triangle inequality. They also gave a polynomial-time algorithm with approximation ratio two when the triangle inequality is satisfied.

When k is restricted, the following results for the D-dimensional Euclidean space are known. For the case $k = 3$, one can solve the max-min 3-dispersion problem in $O(n^2 \log n)$ time [18]. For $k = 2$, the 2-dispersion of P corresponds to the computation of the diameter of P, and one can compute it in $O(n \log n)$ time [19].

In this paper we consider the case where P is a set of points in convex position.

and d is the Euclidean distance. See an example of a 3-dispersion of P in Figure 1. By the brute force algorithm and the algorithm in [18] one can compute a 3-dispersion of P in $O(n^3)$ and $O(n^2 \log n)$ time, respectively, for a set of points on the plane. In this paper we give an algorithm to compute a 3-dispersion of P in $O(n^2)$ time using the property that P is a set of points in convex position.

2 Preliminaries

Let P be a set of n points in convex position on the plane. In this paper, we assume $n \geq 3$. We denote the Euclidean distance between two points u, v by $d(u, v)$. The cost of a set $S \subset P$ is defined as $\mathrm{cost}(S) = \min_{u,v \in S} d(u, v)$. Let $\mathcal{S}_3$ be the set of all possible three points in P. We say $S \in \mathcal{S}_3$ is a 3-dispersion of P if $\mathrm{cost}(S) = \max_{S' \in \mathcal{S}_3} \mathrm{cost}(S')$.

We have the following two lemmas.

▶ **Lemma 2.1.** *If a triangle with corner points p_i, p_r, p_ℓ satisfies $d(p_i, p_r) \geq L$, $d(p_i, p_\ell) \geq L$ and $d(p_\ell, p_r) < L$ for some L, then $\angle p_\ell p_i p_r < 60°$.*

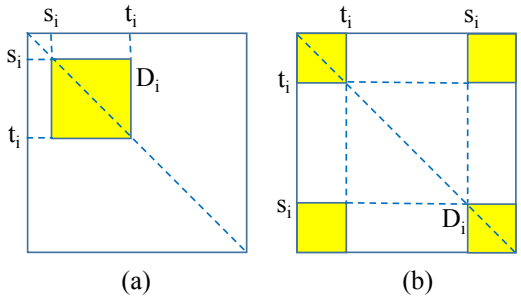


Figure 2 Illustrations for the square submatrix D_i of D for p_i.

▶ **Lemma 2.2.** *If a triangle with corner points p_i, p_r, p_ℓ satisfies $d(p_i, p_r) < L$, $d(p_i, p_\ell) < L$ and $d(p_\ell, p_r) \geq L$ for some L, then $\angle p_\ell p_i p_r > 60°$.*

3 Algorithm

Let $P = (p_1, p_2, \ldots, p_n)$ be the set of points in convex position and assume that they appear clockwise in this order. Let D be the distance matrix of the points in P, that is, the element at row y and column x is $d(p_y, p_x)$. Let $C_1 = \{d(p_i, p_j) \mid 1 \leq i < j \leq n\}$. The cost of a 3-dispersion in P is the distance between some pair of points in P, so it is in C_1.

The outline of our algorithm is as follows. Our algorithm is a binary search and proceeds in at most $2 \log n$ stages. For each stage $j = 1, 2, \ldots, k$, where k is at most $2 \log n$, we (1) compute the median r_j of C_j, where C_j is a subset of C_{j-1}, which is computed in the $(j-1)$st stage (except the case of $j = 1$), (2) compute n square submatrices of D defined by r_j along the main diagonal in D, then (3) we check if some square submatrix among them has an element greater than or equal to r_j, or not. We prove later that at least one square submatrix above has an element greater than or equal to r_j if and only if P has a 3-dispersion with cost r_j or more. If the answer of (3) is YES then we set C_{j+1} as the subset of C_j consisting of the distances greater than or equal to r_j, otherwise we set C_{j+1} as the subset of C_j consisting of the distances less than r_j. Note that in either case the cost of a 3-dispersion of P is in C_{j+1} and $|C_{j+1}| \leq |C_j|/2$ holds. Since the size of C_{j+1} is at most half of C_j and $|C_1| \leq n^2$, the number of stages is at most $\log n^2 = 2 \log n$.

Now, we explain the details of each stage. For the computation of the median in (1), we simply use a linear-time median-finding algorithm [22].

Next, we explain the detail of (2) for each stage j. Given r_j, for each $p_i \in P$, we compute the first point, say $s_i \in P$, on the convex polygon with $d(p_i, s_i) \geq r_j$ when we check the points clockwise from p_i. Similarly, we compute the first point, say $t_i \in P$, on the convex polygon with $d(p_i, t_i) \geq r_j$ when we check the points counterclockwise from p_i. See such an example in Figure 3. Note that, when we check the points clockwise from s_i to t_i, for some point, say p_c, $d(p_i, p_c) < r_j$ may hold. See Figure 3. For each p_i we define a square submatrix D_i of D induced by the rows $s_i, \ldots, t_i$ and the columns $s_i, \ldots, t_i$. See Figure 2(a). Note that D_i is located in D along the main diagonal. The square submatrix D_i may appear in D as four separated squares if it contains p_1 on the clockwise contour from s_i to t_i. See Figure 2(b).

If we search each s_i independently by scanning then total running time for the search of $s_1, s_2, \ldots, s_n$ is $O(n^2)$ in each stage, and $O(n^2 \log n)$ in the whole algorithm. We are going to improve this. Since s_{i+1} may appear before s_i on the clockwise contour (See Figure 4)

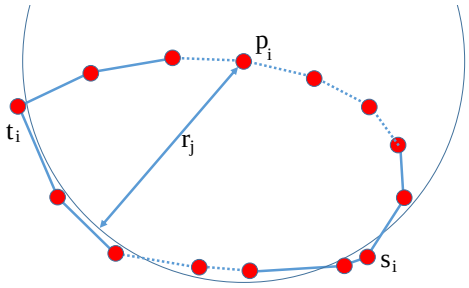


Figure 3 An example of s_i and t_i for p_i. The drawn circle is a circle with the center of p_i the radius of length r_i.

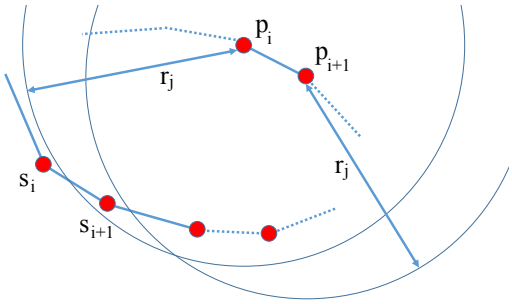


Figure 4 s_{i+1} may appear before s_i on the clockwise contour.

the search is not so simple.

We estimate the total number of distance checks for computing s_i of p_i for each $i = 1, 2, \ldots, n$ in stage 1. Given r_1, we check each point clockwise starting at p_i, and s_i is the first point from p_i which has the distance r_1 or more. It can be observed that the total number of checks for the distance in stage 1 is at most $n + |C_1|/2 \leq n + n^2/2$. In the estimation, n checks are required for the pairs of (s_i, p_i) for every $i = 1, 2, \ldots, n$ and $|C_1|/2$ checks are required for the pairs (p, p_i) which satisfies that p appears between p_i and s_i clockwise and $d(p, p_i) < r_1$, for every $i = 1, 2, \ldots, n$. Remember that r_1 is the median of distances in C_1. Then, in each stage $j = 2, 3, \ldots, k$ ($k \leq 2 \log n$), given r_j, if the answer to (3) of the preceding stage $j - 1$ is YES then we check each point clockwise starting at s_i of the preceding stage $j - 1$ (since $r_j > r_{j-1}$ holds, all points before s_i of the preceding stage are within distance r_j from p_i), otherwise we check each point clockwise starting again at the starting point of the preceding stage $j - 1$. In either case we check at most $n + n^2/2^j$ points in total for the search for $s_1, s_2, \ldots, s_n$ in the stage j, and at most $2n \log n + 2n^2$ points for the whole algorithm. Note that the total number of checks in each stage j is n for $s_1, s_2, \ldots, s_n$ plus $|C_j|/2 \leq n^2/2^j$ for the points with distance less than r_j from its p_i.

Now, we give a lemma mentioned in (3). Assume that we are at stage j, and s_i and t_i for p_i are given. If there is a set of three points in P containing p_i with cost r_j or more, then the square submatrix D_i has an element greater than or equal to r_j. The reverse may be wrong. If the submatrix D_i for some p_i has an element greater than or equal to r_j at row y and column x, it only ensures $d(p_x, p_y) \geq r_j$. That is, $d(p_i, p_x) < r_j$ and/or $d(p_i, p_y) < r_j$ may hold. We show that this situation cannot occur in the following lemma.

▶ **Lemma 3.1.** *The square submatrix D_i of stage j has an element greater than or equal to*

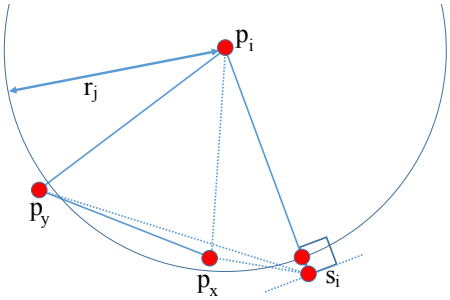


Figure 5 An illustration for Lemma 3.1.

r_j if and only if there is a set of three points $S \subset P$ including p_i with $\mathrm{cost}(S) \geq r_j$.

Proof. If there is a set of three points $S \subset P$ including p_i with $\mathrm{cost}(S) \geq r_j$ then clearly the square submatrix D_i of stage j has an element greater than or equal to r_j.

We only prove the other direction, that is, if the square submatrix D_i of stage j has an element greater than or equal to r_j then there is a set of three points $S \subset P$ including p_i with $\mathrm{cost}(S) \geq r_j$. Assume that D_i has an element greater than or equal to r_j at row y and column x, that is $d(p_x, p_y) \geq r_j$. We have the following four cases and in each case we show that there exists a set S of three points such that $\mathrm{cost}(S) \geq r_j$.

Case 1: $d(p_i, p_x) \geq r_j$ and $d(p_i, p_y) \geq r_j$.
The set $S = \{p_i, p_x, p_y\}$ has $\mathrm{cost}(S) \geq r_j$.

Case 2: $d(p_i, p_x) < r_j$ and $d(p_i, p_y) < r_j$.
We show that, for $S = \{p_i, s_i, t_i\}$, $\mathrm{cost}(S) \geq r_j$ holds. We assume for a contradiction that $d(s_i, t_i) < r_j$ holds. Then, we have $\angle s_i p_i t_i < 60°$ by Lemma 2.1 and $\angle p_x p_i p_y > 60°$ by Lemma 2.2. This is a contradiction to the convexity of P.

Case 3: $d(p_i, p_x) < r_j$ and $d(p_i, p_y) \geq r_j$.
In this case, we show that the set $\{p_i, s_i, p_y\}$ attains $\mathrm{cost}(S) \geq r_j$. Since, $d(p_i, p_y) \geq r_j$ and $d(p_i, s_i) \geq r_j$, we have to prove $d(s_i, p_y) \geq r_j$.

Assume for a contradiction that $d(s_i, p_y) < r_j$ holds. See Figure 5. Now, we first show that $\{s_i, p_x, p_y\}$ forms an obtuse triangle with the obtuse angle p_x, below. We focus on the rectangle consisting of p_i, p_y, p_x, and s_i. Since $d(p_i, p_y) \geq r_j$ and $d(p_i, s_i) \geq r_j$, and $d(s_i, p_y) < r_j$, we have $\angle s_i p_i p_y < 60°$ by Lemma 2.1. Let p' be the point on the line segment between p_i and s_i with $d(p_i, p') = r_j$. Since $\angle p_i p' p_x < 90°$ holds, we can observe that $\angle p_i s_i p_x < 90°$ holds. Since $d(p_i, p_y) \geq r_j$, $d(p_x, p_y) \geq r_j$, and $d(p_i, p_x) < r_j$, we have $\angle p_i p_y p_x < 60°$ by Lemma 2.1. Now, the sum of the internal angles of the quadrangle consisting of p_i, s_i, p_x, and p_y implies that $\angle s_i p_x p_y \geq 150°$, and $\{s_i, p_x, p_y\}$ are the points of an obtuse triangle with obtuse angle at p_x. However $d(p_x, p_y) \geq r_j$ and $d(s_i, p_y) < r_j$, which is a contradiction.

Case 4: $d(p_i, p_x) \geq r_j$ and $d(p_i, p_y) < r_j$.
Symmetry to **Case 3**. Omitted. □

Now, we are ready to describe our algorithm and the estimation of the running time. First, as a preprocessing, we construct the set $C_1 = \{d(p_i, p_j) \mid 1 \leq i < j \leq n\}$ of distances and $n \times n$ distance matrix D. Next, we repeat the following stage for each $j = 1, 2, \ldots, k$, where $k \leq 2 \log n$. (1) we compute the median r_j of C_j, (2) compute s_i and t_i of p_i for

$i = 1, 2, \ldots, n$, and (3) check whether there exists an index i, $(1 \leq i \leq n)$, such that the maximum value of D_i is greater than or equal to r_j. Then, if such i exists, we set $C_{j+1} = \{d(p_i, p_j) \in C_j \mid d(p_i, p_j) \geq r_j\}$, otherwise, we set $C_{j+1} = \{d(p_i, p_j) \in C_j \mid d(p_i, p_j) < r_j\}$.

The analysis of the running time is as follows. The preprocessing can be done in $O(n^2)$ time. For (1), we can compute the median r_j of stage j in $O(n/2^{j-1})$ time by using a linear-time median-finding algorithm [22], and hence $O(n^2)$ time for the whole algorithm. The computation for (2) can be done in $O(n^2)$ time in the whole algorithm, as described above. For (3), after $O(n^2)$-time preprocessing for D, we can compute the maximum element in the given submatrix in D in $O(1)$ time for each query by using the range-query algorithm [25], so we need $O(n)$ time for each stage and $O(n \log n)$ time for the whole algorithm. (For a separated square as shown in Figure 2(b) we need four queries but total time is still a constant.)

Now, we have our main theorem.

▶ **Theorem 3.2.** *One can compute a 3-dispersion of P in $O(n^2)$ time if P is a set of n points in convex position.*

Acknowledgments. This work was supported by JSPS KAKENHI Grant Numbers JP18H04091, JP19K11812, JP20H05793, JP20H05962, JP20K19742. The fourth author is also supported by JST CREST Grant Number JPMJCR1401.

References

1 P. Agarwal and M. Sharir. Efficient algorithms for geometric optimization. *ACM Comput. Surv.*, 30:412–458, 1998.

2 T. Akagi, T. Araki, T. Horiyama, S. Nakano, Y. Okamoto, Y. Otachi, T. Saitoh, R. Uehara, T. Uno, and K. Wasa. Exact algorithms for the max-min dispersion problem. *Proc. of FAW 2018*, LNCS 10823:263–272, 2018.

3 T. Akagi and S. Nakano. Dispersion on the line. *IPSJ SIG Technical Reports*, 2016-AL-158-3, 2016.

4 K. Amano and S. Nakano. An approximation algorithm for the 2-dispersion problem. *IEICE TRANS. INF.SYST.*, E103-D:506–508, 2020.

5 T. Araki and S. Nakano. The max-min dispersion on a line. *Proc. of COCOA 2018*, LNCS 11346:672–678, 2018.

6 C. Baur and S. P. Fekete. Approximation of geometric dispersion problems. *Proc. of APPROX 1998*, pages 63–75, 1998.

7 B. Birnbaum and K. J. Goldman. An improved analysis for a greedy remote-clique algorithm using factor-revealing LPs. *Algorithmica*, 50:42–59, 2009.

8 A. Cevallos, F. Eisenbrand, and R. Zenklusen. Max-sum diversity via convex programming. *Proc. of SoCG 2016*, pages 26:1–26:14, 2016.

9 A. Cevallos, F. Eisenbrand, and R. Zenklusen. Local search for max-sum diversification. *Proc. of SODA 2017*, pages 130–142, 2017.

10 B. Chandra and M. M. Halldorsson. Approximation algorithms for dispersion problems. *J. of Algorithms*, 38:438–465, 2001.

11 Z. Drezner. *Facility location: A Survey of Applications and Methods.* Springer, 1995.

12 Z. Drezner and H.W. Hamacher. *Facility Location: Applications and Theory.* Springer, 2004.

13 E. Erkut. The discrete p-dispersion problem. *European Journal of Operational Research*, 46:48–60, 1990.

14 E. Erkut, Y. Ulkusal, and O. Yenicerioglu. A comparison of p-dispersion heuristics. *Computers & Operational Research*, 21:1103–1113, 1994.

15 S. P. Fekete and H. Meijer. Maximum dispersion and geometric maximum weight cliques. *Algorithmica*, 38:501–511, 2004.

16 G. Frederickson. Optimal algorithms for tree partitioning. *Proc. of SODA 1991*, pages 168–177, 1991.

17 R. Hassin, S. Rubinstein, and A. Tamir. Approximation algorithms for maximum dispersion. *Operation Research Letters*, 21:133–137, 1997.

18 T. Horiyama, S. Nakano, T. Saitoh, K. Suetsugu, A. Suzuki, R. Uehara, T. Uno, and K. Wasa. Max-min 3-dispersion problems. *Proc. of COCOON 2019*, LNCS 11653:291–300, 2019.

19 F. P. Preparata and M. I. Shamos. *Computational geometry: an introduction*. Springer-Verlag, 1985.

20 S. S. Ravi, D. J. Rosenkrantz, and G. K. Tayi. Heuristic and special case algorithms for dispersion problems. *Operations Research*, 42:299–310, 1994.

21 M. Sydow. Approximation guarantees for max sum and max min facility dispersion with parameterised triangle inequality and applications in result diversification. *Mathematica Applicanda*, 42:241–257, 2014.

22 R. L. Rivest T. H. Cormen, C. E. Leiserson and C. Stein. *Introduction to algorithms, Third Edition*. MIT Press, 2000.

23 K. H. Tsai and D. W. Wang. Optimal algorithms for circle partitioning. *Proc. of COCOON 1997*, LNCS 1276:304–310, 1997.

24 D. W. Wang and Y.-S. Kuo. A study on two geometric location problems. *Information Processing Letters*, 28:281–286, 1988.

25 H. Yuan and M. J. Atallah. Data structures for range minimum queries in multidimensional arrays. *Proc. of SODA 2010*, pages 150–160, 2010.

The maximal number of 3-term arithmetic progressions in finite sets in different geometries

Itai Benjamini[1] **and Shoni Gilboa**[2]

1 **Department of Mathematics, Weizmann Institute of Science, Rehovot 7610001, Israel**
2 **Department of Mathematics and Computer Science, The Open University of Israel, Raanana 4353701, Israel**

Abstract

Green and Sisask showed that the maximal number of 3-term arithmetic progressions in n-element sets of integers is $\lceil n^2/2 \rceil$; it is easy to see that the same holds if the set of integers is replaced by the real line or by any Euclidean space. We study this problem in general metric spaces, where a triple (a, b, c) of points in a metric space is considered a 3-*term arithmetic progression* if $d(a, b) = d(b, c) = \frac{1}{2}d(a, c)$. In particular, we show that the result of Green and Sisask extends to any Cartan–Hadamard manifold (in particular, to the hyperbolic spaces), but does not hold in spherical geometry or in the r-regular tree, for any $r \geq 3$.

Related Version arXiv:2011.04410

1 Introduction

It was shown in [6, Theorem 1.2] that the maximal number of 3-term arithmetic progressions in n-element sets of integers is $\lceil n^2/2 \rceil$ (counting increasing, decreasing and constant progressions). Combined with some tools from additive combinatorics, this result was used in [6] to obtain their main result that $\lceil n^2/2 \rceil$ is also the maximal number of 3-term arithmetic progressions in n-element subsets of the additive group $\mathbb{Z}/p\mathbb{Z}$ for prime p, provided that n/p is smaller than some absolute constant. Additive structure is probably the most natural context of arithmetic progressions, but it may be viewed also as a metric notion, which is the direction we pursue here; we study the maximal number of 3-term arithmetic progressions in n-element subsets of various metric spaces and examine how it relates to geometric properties of these spaces.

▶ **Definition 1.1.** Let M be a metric space. We say that $(a, b, c) \in M^3$ is a 3-*term arithmetic progression* in M if $d_M(a, b) = d_M(b, c) = \frac{1}{2}d_M(a, c)$, where d_M is the metric of M. For any set $A \subseteq M$, let

$$\mathcal{AT}_M(A) := \{(a, b, c) \in A^3 \mid d_M(a, b) = d_M(b, c) = \tfrac{1}{2}d_M(a, c)\}$$

be the set of 3-term arithmetic progressions in the set A. For every positive integer n, let

$$\mu_n(M) := \max\{|\mathcal{AT}_M(A)| : A \subseteq M,\ |A| = n\}$$

be the maximal number of 3-term arithmetic progressions in n-element subsets of M.

▶ **Observation 1.2.** *Note that $(b, b, b) \in \mathcal{AT}_M(A)$ for every $b \in A$ and that if $(a, b, c) \in \mathcal{AT}_M(A)$, then $(c, b, a) \in \mathcal{AT}_M(A)$ as well.*

As already mentioned, it was shown in [6] that $\mu_n(\mathbb{Z}) = \lceil n^2/2 \rceil$ for every n, and the same argument shows that for every n,

$$\mu_n(\mathbb{R}) = \lceil n^2/2 \rceil. \tag{1}$$

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

This yields, by a simple projection argument, that the same is true for Euclidean spaces of any dimension. We show that this extends to a rather large class of metric spaces. First, let us recall some basic notions. Let M be a metric space; a curve $\gamma : I \to M$, where I is a connected subset of the real line, is a *geodesic* if $d_M(\gamma(y), \gamma(x)) = y - x$ for every $x < y$ in I; a set $\Gamma \subseteq M$ is a *geodesic segment with endpoints* p, q if there is a geodesic $\gamma : [a, b] \to M$ such that $\Gamma = \gamma([a, b])$, $p = \gamma(a)$ and $q = \gamma(b)$; the metric space M is *uniquely geodesic* if any two distinct points in M are the endpoints of a unique geodesic segment; finally, a curve $\gamma : I \to M$, where I is a connected subset of the real line, is a *local geodesic* if around every $a \in I$ there is an open interval I_a such that the restriction of γ to $I \cap I_a$ is a geodesic.

▶ **Theorem 1.3.** *Let M be a uniquely geodesic Riemannian manifold in which every local geodesic is a geodesic. Then, $\mu_n(M) = \lceil n^2/2 \rceil$ for every n; moreover, any set A of n points in M for which $|\mathcal{AT}_M(A)| = \lceil n^2/2 \rceil$ is contained in the image of a geodesic.*

The proof of Theorem 1.3 is given in section 2.

▶ Remark. In particular, Theorem 1.3 applies to the hyperbolic spaces, and more generally, to any *Cartan–Hadamard manifold*, i.e., complete simply connected Riemannian manifold that has everywhere nonpositive sectional curvature (see, e.g., [1, 4]). However, the result does not extend to the wider class of *Hadamard spaces*, i.e., complete metric spaces of global nonpositive curvature, in the sense of A. D. Alexandrov (note that each such metric space is uniquely geodesic, and every local geodesic in it is a geodesic; see, e.g., [2, 3]). For instance, let $\mathbb{T}_r$ be the (discrete) r-regular tree, $r \geq 2$, equipped with the graph metric, and let $\hat{\mathbb{T}}_r$ be the corresponding metric graph, where all the edges have unit length, which is a Hadamard space. A simple computation shows that for every ball A in $\mathbb{T}_r$,

$$|\mathcal{AT}_{\mathbb{T}_r}(A)| = \left(\frac{1}{2} + \frac{(r-2)^2}{2r^2}\right)|A|^2 + \frac{2(r-2)}{r^2}|A| + \frac{2}{r^2}.$$

Since obviously $\mu_n(\hat{\mathbb{T}}_r) \geq \mu_n(\mathbb{T}_r)$ for every n, it follows that for every $r \geq 3$,

$$\limsup_{n\to\infty} \frac{\mu_n(\hat{\mathbb{T}}_r)}{n^2} \geq \limsup_{n\to\infty} \frac{\mu_n(\mathbb{T}_r)}{n^2} \geq \frac{1}{2} + \frac{(r-2)^2}{2r^2} > \frac{1}{2} = \lim_{n\to\infty} \frac{\lceil n^2/2 \rceil}{n^2}.$$

Next, we consider the unit circle $S^1 = \{u \in \mathbb{R}^2 : |u| = 1\}$, with respect to the arc length metric.

▶ **Theorem 1.4.** *For every $n \neq 2$,*

$$\mu_n(S^1) = \frac{1}{2}n^2 + \begin{cases} n & n \bmod 4 = 0, \\ \frac{1}{2}n & n \bmod 4 = 1, \\ 2 & n \bmod 4 = 2, \\ \frac{1}{2}n - 1 & n \bmod 4 = 3. \end{cases}$$

The proof of Theorem 1.4 is given in section 3.

By considering subsets of the unit sphere $S^2 = \{u \in \mathbb{R}^3 : |u| = 1\}$ that are composed of an appropriate set of $n - 2$ points on a *great circle* of the sphere and the pair of respective *poles*, it is simple to show that for every $n \geq 2$,

$$\mu_n(S^2) \geq \frac{1}{2}n^2 + \begin{cases} 2n - 4 & n \bmod 4 = 0, \\ \frac{5}{2}n - 8 & n \bmod 4 = 1, \\ 3n - 6 & n \bmod 4 = 2, \\ \frac{5}{2}n - 7 & n \bmod 4 = 3. \end{cases} \tag{2}$$

We believe that this lower bound for $\mu_n(S^2)$ is tight for every $n \geq 2$. Note that combining (2) with Theorem 1.4 yields that $\mu_n(S^2) > \mu_n(S^1)$ for every $n \geq 5$.

2 Proof of Theorem 1.3

Theorem 1.3 will follow from the following, more general, theorem.

▶ **Theorem 2.1.** *Let M be a metric space, and let $\mathcal{L}$ be a family of subsets of M; we will refer to any set in $\mathcal{L}$ as a 'line'. Assume that the following conditions hold:*

1. *Each line in $\mathcal{L}$ is isometric to a subset of the real line.*
2. *For any two distinct points in M there is a unique line in $\mathcal{L}$ containing them both.*
3. *For every nonconstant 3-term arithmetic progression (a, b, c) in M, the points a, b, c lie on a common line in $\mathcal{L}$ (which is obviously unique, by the previous condition).*

Then, $\mu_n(M) \leq \lceil n^2/2 \rceil$ for every n; moreover, if $\mu_n(M) = \lceil n^2/2 \rceil$, then any set A of n points in M for which $|\mathcal{AT}_M(A)| = \lceil n^2/2 \rceil$ is contained in a line in $\mathcal{L}$.

Proof. Let A be a set of n points in M.

If A is contained in a line $L \in \mathcal{L}$, then since L is isometric to a subset of the real line, it follows from (1) that $|\mathcal{AT}_M(A)| \leq \mu_n(L) \leq \mu_n(\mathbb{R}) = \lceil n^2/2 \rceil$.

Assume that A is not contained in a line in $\mathcal{L}$. For every $L \in \mathcal{L}$, let $r_L := |A \cap L|$. Let $\mathcal{L}_A := \{L \in \mathcal{L} \mid r_L \geq 2\}$ be the set of lines 'determined' by the set A. For every $L \in \mathcal{L}_A$, since L is isometric to a subset of the real line, it follows from (1) that

$$|\mathcal{AT}_M(A \cap L)| - r_L \leq \mu_{r_L}(\mathbb{R}) - r_L = \left\lceil \frac{r_L^2}{2} \right\rceil - r_L = \binom{r_L}{2} - \left\lfloor \frac{r_L}{2} \right\rfloor \leq \binom{r_L}{2} - 1. \quad (3)$$

For any two distinct points of A, there is a unique line in $\mathcal{L}_A$ containing them both. Therefore,

$$\sum_{L \in \mathcal{L}_A} \binom{r_L}{2} = \binom{n}{2}, \quad (4)$$

and moreover, since the points in A are not all on a single line, it follows from Fisher's inequality [5] that

$$|\mathcal{L}_A| \geq n. \quad (5)$$

For each nonconstant $(a, b, c) \in \mathcal{AT}_M(A)$, there is a unique line in $\mathcal{L}_A$ containing all three points a, b, c. Hence,

$$|\mathcal{AT}_M(A)| - n = \sum_{L \in \mathcal{L}_A} \left(|\mathcal{AT}_M(A \cap L)| - r_L \right).$$

Therefore, by (3), (4) and (5),

$$|\mathcal{AT}_M(A)| - n \leq \sum_{L \in \mathcal{L}_A} \left(\binom{r_L}{2} - 1 \right) = \binom{n}{2} - |\mathcal{L}_A| \leq \binom{n}{2} - n,$$

and hence, $|\mathcal{AT}_M(A)| \leq \binom{n}{2} < \lceil n^2/2 \rceil$, which concludes the proof. ◀

We proceed to prove Theorem 1.3. Consider the family of 'lines'

$$\mathcal{L} := \{\gamma(I) \mid \gamma : I \to M \text{ is a maximal geodesic}\},$$

where a geodesic is maximal if it cannot be extended to a geodesic with a larger domain.

Any geodesic segment in M is contained in a unique line in $\mathcal{L}$. Indeed, let Γ be a geodesic segment in M, let p, q be its endpoints and let $\delta := d_M(p, q)$. There is a unique geodesic $\gamma : [0, \delta] \to M$ such that $\gamma([0, \delta]) = \Gamma$, $\gamma(0) = p$ and $\gamma(\delta) = q$. Since M is a Riemannian manifold, γ may be uniquely extended to a maximal local geodesic $\hat{\gamma} : I \to M$ (i.e., a local geodesic that cannot be extended to a local geodesic with a larger domain). Since each local geodesic in M is a geodesic, it follows that $\hat{\gamma}(I)$ is the unique line in $\mathcal{L}$ containing the geodesic segment Γ.

Let us verify that the metric space M and the family $\mathcal{L}$ of subsets of M satisfy all the conditions of Theorem 2.1. For any geodesic $\gamma : I \to M$, the set $\gamma(I)$ is isometric to the subset I of the real line; in particular, every line in $\mathcal{L}$ is isometric to a subset of the real line. For any two distinct points p, q in M there is a unique geodesic segment Γ with endpoints p, q, since M is uniquely geodesic; the unique line in $\mathcal{L}$ containing Γ is obviously the unique line in $\mathcal{L}$ containing both p and q. Finally, if (a, b, c) is a nonconstant 3-term arithmetic progression in M, then it is straightforward to show, since $d_M(a, b) + d_M(b, c) = d_M(a, c)$, that the union Γ of a geodesic segment with endpoints a, b and a geodesic segment with endpoints b, c is necessarily a geodesic segment with endpoints a, c; hence, the points a, b, c lie on the line in $\mathcal{L}$ containing Γ.

Theorem 1.3 now follows from Theorem 2.1, upon taking an arbitrary nonconstant geodesic $\gamma : I \to M$ and observing that for every n we may find an n-term arithmetic progression $A_n \subset I$ and then, $\mu_n(M) \geq \mu_n(\gamma(I)) = \mu_n(I) \geq |\mathcal{AT}_{\mathbb{R}}(A_n)| = \lceil n^2/2 \rceil$.

3 Proof of Theorem 1.4

For every pair of distinct points a, b in S^1, let $C_{a,b}$ be the open arc of S^1 from a to b counterclockwise, and let $M_{a,b}$ be the midpoint of this arc.

First, we prove the lower bound in Theorem 1.4. We say that a set $\{p_1, p_2, \ldots, p_n\}$ of $n \geq 2$ points in S^1, where the points $p_1, p_2, \ldots, p_n$ are ordered counterclockwise, is *evenly spread around the circle* if $d_{S^1}(p_1, p_2) = \cdots = d_{S^1}(p_{n-1}, p_n) = d_{S^1}(p_n, p_1) = 2\pi/n$. Let $\mathcal{F}_1 := \{\{a\} \mid a \in S^1\}$ and for every $n \geq 2$, let $\mathcal{F}_n$ be the family of all n-element subsets of S^1 that are evenly spread around the circle. For every positive integer n which is divisible by 4, let ρ_n be the rotation of S^1 by an angle of π/n (counterclockwise) and let

$$\begin{aligned}
\mathcal{F}_n^{[-1]} &:= \{A \setminus \{a\} \mid A \in \mathcal{F}_n,\ a \in A\},\\
\mathcal{F}_n^{[-2]} &:= \{A \setminus \{a, b\} \mid A \in \mathcal{F}_n,\ a, b \in A \text{ such that } d_{S^1}(a, b) \leq \tfrac{\pi}{2} \text{ and } M_{a,b}, M_{b,a} \in A\},\\
\mathcal{F}_n^{[+1]} &:= \{A \cup \{a\} \mid A \in \mathcal{F}_n,\ \rho_n(a) \in A\},\\
\mathcal{F}_n^{[+2]} &:= \{A \cup \{a, b\} \mid A \in \mathcal{F}_n,\ a, b \in S^1 \text{ such that } \rho_n(a), \rho_n(b), M_{a,b}, M_{b,a} \in A\}.
\end{aligned}$$

A straightforward computation yields that for every positive integer n,

$$|\mathcal{AT}_{S^1}(A)| = 2n\lfloor n/4 \rfloor + n \text{ for any } A \in \mathcal{F}_n, \tag{6}$$

and for every positive integer n which is divisible by 4,

$$|\mathcal{AT}_{S^1}(A)| = \tfrac{1}{2}(n-1)^2 + \tfrac{1}{2}(n-1) - 1 \quad \text{for any } A \in \mathcal{F}_n^{[-1]}, \tag{7a}$$

$$|\mathcal{AT}_{S^1}(A)| = \tfrac{1}{2}(n-2)^2 + 2 \quad \text{for any } A \in \mathcal{F}_n^{[-2]}, \tag{7b}$$

$$|\mathcal{AT}_{S^1}(A)| = \tfrac{1}{2}(n+1)^2 + \tfrac{1}{2}(n+1) \quad \text{for any } A \in \mathcal{F}_n^{[+1]}, \tag{7c}$$

$$|\mathcal{AT}_{S^1}(A)| = \tfrac{1}{2}(n+2)^2 + 2 \quad \text{for any } A \in \mathcal{F}_n^{[+2]}. \tag{7d}$$

The lower bound in Theorem 1.4 follows since if $n \bmod 4 = 0$, then $|\mathcal{AT}_{S^1}(A)| = \frac{1}{2}n^2 + n$ for any $A \in \mathcal{F}_n$, by (6); if $n \bmod 4 = 1$, then $|\mathcal{AT}_{S^1}(A)| = \frac{1}{2}n^2 + \frac{1}{2}n$ for any $A \in \mathcal{F}_n \cup \mathcal{F}_{n-1}^{[+1]}$, by (6) and (7c); if $n \bmod 4 = 2$ and $n > 2$, then $|\mathcal{AT}_{S^1}(A)| = \frac{1}{2}n^2 + 2$ for any $A \in \mathcal{F}_{n+2}^{[-2]} \cup \mathcal{F}_{n-2}^{[+2]}$, by (7b) and (7d); finally, if $n \bmod 4 = 3$, then $|\mathcal{AT}_{S^1}(A)| = \frac{1}{2}n^2 + \frac{1}{2}n - 1$ for any $A \in \mathcal{F}_{n+1}^{[-1]}$, by (7a).

We proceed to prove the upper bound in Theorem 1.4. Let A be a set of n points in S^1. For any $b \in A$, denote

$$w_A(b) := |\{(x, y, z) \in \mathcal{AT}_{S^1}(A) : y = b\}|.$$

Note that $w_A(b)$ is odd for every $b \in A$, by Observation 1.2; hence, $\frac{1}{2}w_A(a) + \frac{1}{2}w_A(b)$ is an integer for every $a, b \in A$. Denote

$$\mathcal{P} := \left\{\{a, b\} \subseteq A \mid a \neq b,\ \left||A \cap C_{a,b}| - |A \cap C_{b,a}|\right| \leq 1\right\}.$$

▶ **Lemma 3.1.** *For any $\{a, b\} \in \mathcal{P}$,*

$$\tfrac{1}{2}w_A(a) + \tfrac{1}{2}w_A(b) \leq \left\lfloor \tfrac{n}{2} \right\rfloor + 1. \tag{8}$$

Proof. Denote

$$H_a := \{x \in S^1 \mid d_{S^1}(a, x) \leq \tfrac{\pi}{2}\}, \quad H_b := \{x \in S^1 \mid d_{S^1}(b, x) \leq \tfrac{\pi}{2}\}.$$

We may assume that the arc $C_{a,b}$ is at least as long as the arc $C_{b,a}$, and that if the two arcs have the same length then $|A \cap C_{a,b}| \leq |A \cap C_{b,a}|$.

If $(b, b, b) \neq (x, b, y) \in \mathcal{AT}_{S^1}(A)$, then x, y are both in $A \cap H_b$ and at least one of them is in the arc $C_{a,b}$. Hence, $w_A(b) \leq 1 + 2|A \cap H_b \cap C_{a,b}|$. Similarly, $w_A(a) \leq 1 + 2|A \cap H_a \cap C_{a,b}|$ and hence,

$$\tfrac{1}{2}w_A(a) + \tfrac{1}{2}w_A(b) \leq 1 + |A \cap H_a \cap C_{a,b}| + |A \cap H_b \cap C_{a,b}|. \tag{9}$$

If the points a, b are antipodal, then $H_a \cup H_b = S^1$, $H_a \cap H_b = \{M_{a,b}, M_{b,a}\}$, $|A \cap C_{a,b}| = \lfloor \frac{n-2}{2} \rfloor$ and hence,

$$|A \cap H_a \cap C_{a,b}| + |A \cap H_b \cap C_{a,b}| = |A \cap C_{a,b}| + |A \cap \{M_{a,b}\}| \leq \left\lfloor \tfrac{n-2}{2} \right\rfloor + 1 = \left\lfloor \tfrac{n}{2} \right\rfloor. \tag{10}$$

If the points a, b are not antipodal, then the set $C_{a,b} \cap H_a \cap H_b$ is empty and hence

$$|A \cap H_a \cap C_{a,b}| + |A \cap H_b \cap C_{a,b}| \leq |A \cap C_{a,b}| \leq \left\lceil \tfrac{n-2}{2} \right\rceil = \left\lceil \tfrac{n}{2} \right\rceil - 1 \leq \left\lfloor \tfrac{n}{2} \right\rfloor. \tag{11}$$

Combining (9), (10) and (11) yields (8). ◀

▶ **Observation 3.2.** *By examining closely the proof of Lemma 3.1, it follows that if $\{a, b\} \in \mathcal{P}$ and $\frac{1}{2}w_A(a) + \frac{1}{2}w_A(b) = \frac{n}{2} + 1$ (in particular, n is even), then the points a, b are antipodal, the set A is invariant under the reflection of $\mathbb{R}^2$ through the line through the points a, b and moreover, the points $M_{a,b}, M_{b,a}$ are in A.*

If n is odd, then by (8),

$$|\mathcal{AT}_{S^1}(A)| = \sum_{b \in A} w_A(b) = \sum_{\{a,b\} \in \mathcal{P}} \left(\tfrac{1}{2}w_A(a) + \tfrac{1}{2}w_A(b)\right) \leq |\mathcal{P}| \left(\tfrac{n-1}{2} + 1\right) = \tfrac{1}{2}n^2 + \tfrac{1}{2}n,$$

which yields the desired upper bound if $n \bmod 4 = 1$. If $n \bmod 4 = 3$, then the desired upper bound follows since $|\mathcal{AT}_{S^1}(A)| - n$ is even, by Observation 1.2, whereas $\left(\frac{1}{2}n^2 + \frac{1}{2}n\right) - n = n\frac{n-1}{2}$ is odd in this case.

If n is even, then

$$|\mathcal{AT}_{S^1}(A)| = \sum_{b\in A} w_A(b) = 2 \sum_{\{a,b\}\in\mathcal{P}} \left(\tfrac{1}{2}w_A(a) + \tfrac{1}{2}w_A(b)\right). \tag{12}$$

Hence, if $n \bmod 4 = 0$ then $|\mathcal{AT}_{S^1}(A)| \le 2|\mathcal{P}|\left(\frac{n}{2}+1\right) = \frac{1}{2}n^2 + n$, by (8), as desired. Finally, suppose that $n \bmod 4 = 2$ and let

$$\mathcal{P}_0 := \left\{\{a,b\} \in \mathcal{P} \mid \tfrac{1}{2}w_A(a) + \tfrac{1}{2}w_A(b) = \tfrac{n}{2}+1\right\}.$$

If $|\mathcal{P}_0| \le 1$, then by (12) and (8),

$$|\mathcal{AT}_{S^1}(A)| \le 2|\mathcal{P}_0|\left(\tfrac{n}{2}+1\right) + 2\left(|\mathcal{P}| - |\mathcal{P}_0|\right)\tfrac{n}{2} = \tfrac{1}{2}n^2 + 2|\mathcal{P}_0| \le \tfrac{1}{2}n^2 + 2$$

as desired. To conclude the proof we will show that if $r := |\mathcal{P}_0| > 1$, then $|\mathcal{AT}_{S^1}(A)| < \frac{1}{2}n^2+2$. Note that by Observation 3.2, each member of $\mathcal{P}_0$ is a pair of antipodal points. Suppose that $\mathcal{P}_0 = \{\{p_i, p_{r+i}\}\}_{i=0}^{r-1}$, where the points $p_0, p_1, \ldots, p_{2r-1}$ are ordered counterclockwise. It follows from Observation 3.2 that the set $\{p_0, p_1, \ldots, p_{2r-1}\}$ is evenly spread around the circle and moreover, r is odd, since n is not divisible by 4. Let $s := \frac{r-1}{2}$. For every $0 \le i \le 2r-1$, the point

$$a_i := M_{p_i, p_{(i+1) \bmod 2r}} = M_{p_{(i-s) \bmod 2r}, p_{(i+1+s) \bmod 2r}}$$

is in A, by Observation 3.2, since $\{p_{(i-s) \bmod 2r}, p_{(i+1+s) \bmod 2r}\} \in \mathcal{P}_0$. Using Observation 3.2 once more, it follows that there are m_0, m_1 such that for every $0 \le i \le 2r-1$,

$$|A \cap C_{p_i,a_i}| = m_{i \bmod 2}, \quad |A \cap C_{a_i, p_{(i+1) \bmod 2r}}| = m_{(i+1) \bmod 2}.$$

Necessarily $m_0 \ne m_1$, since n is not divisible by 4. Now it is simple to show, by using an argument similar to the one used to prove Lemma 3.1, that for every $0 \le i \le 2r-1$,

$$\tfrac{1}{2}w_A(a_i) + \tfrac{1}{2}w_A(b_i) \le \tfrac{n}{2} - 1,$$

where b_i is the point in A for which $\{a_i, b_i\} \in \mathcal{P}$. Hence, if we denote $\mathcal{P}_1 := \{\{a_i, b_i\}\}_{i=0}^{2r-1}$, then by (12) and (8),

$$\begin{aligned} |\mathcal{AT}_{S^1}(A)| &\le 2|\mathcal{P}_0|\left(\tfrac{n}{2}+1\right) + 2|\mathcal{P}_1|\left(\tfrac{n}{2}-1\right) + 2\left(|\mathcal{P}| - |\mathcal{P}_0| - |\mathcal{P}_1|\right)\tfrac{n}{2} \\ &= \tfrac{1}{2}n^2 + 2\left(|\mathcal{P}_0| - |\mathcal{P}_1|\right) < \tfrac{1}{2}n^2 + 2. \end{aligned}$$

Acknowledgments. We thank Lev Buhovski, Dan Hefetz, Bo'az Klartag and Pierre Pansu for fruitful discussions.

References

1 Werner Ballmann, Mikhael Gromov, and Viktor Schroeder. *Manifolds of nonpositive curvature*, volume 61 of *Progress in Mathematics*. Birkhäuser Boston, Inc., Boston, MA, 1985. doi:10.1007/978-1-4684-9159-3.

2 Martin R. Bridson and André Haefliger. *Metric spaces of non-positive curvature*, volume 319 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, 1999. doi:10.1007/978-3-662-12494-9.

3 Dmitri Burago, Yuri Burago, and Sergei Ivanov. *A course in metric geometry*, volume 33 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 2001. doi:10.1090/gsm/033.

4 Jeff Cheeger and David G. Ebin. *Comparison theorems in Riemannian geometry*. North-Holland Publishing Co., Amsterdam-Oxford; American Elsevier Publishing Co., Inc., New York, 1975. North-Holland Mathematical Library, Vol. 9.

5 R. A. Fisher. An examination of the different possible solutions of a problem in incomplete blocks. *Ann. Eugenics*, 10:52–75, 1940.

6 Ben Green and Olof Sisask. On the maximal number of 3-term arithmetic progressions in subsets of $\mathbb{Z}/p\mathbb{Z}$. *Bull. Lond. Math. Soc.*, 40(6):945–955, 2008. doi:10.1112/blms/bdn074.

Notes on pivot pairings

Barbara Giunti[1,2]

1 **Università degli Studi di Modena e Reggio Emilia, Italy**

2 **Graz University of Technology, Graz, Austria**
bgiunti@tugraz.at

Abstract

We present a row reduction algorithm to compute the barcode decomposition of persistence modules. This algorithm dualises the standard persistence algorithm and clarifies the symmetry between clear and compress optimisations.

1 Introduction

Persistent homology is a tool of Topological Data Analysis (TDA) whose applications range widely from biology to urban planning to neuroscience (see [16] for an extended list of applications). Persistent homology summarises a dataset's information in the form of barcode [6, 15]. The efficient computation of such barcodes is one of the main algorithmic problems in TDA [17].

We provide an algorithm (Algorithm 1) for the barcode decomposition based on row pivot pairing, which is the dual of the column pivot pairing presented in [10, 13]. The algorithm reduces the boundary matrix of a given filtered simplicial complex proceeding by rows and performing row additions, allowing the full exploitation of the compress optimisation [3].

A non-exhaustive list of algorithms to decompose persistence modules into interval modules includes the so-called standard algorithm [13], the chunk algorithm [3], the twist algorithm [9], the pHrow algorithm [12], and Ripser [2]. The implementation of the first four can be found in [4] (cf. [5]), and that of Ripser in [1]. All of them take as input a filtered simplicial complex and employ column operations. Moreover, in a recent and independent work [14], a dualisation of the standard algorithm is presented, using row operations. In addition to these algorithms, in [7] it is shown that the barcode decomposition can also be achieved via the decomposition of filtered chain complexes, whose reduction is performed by row operations. However, the reduction in [7] is different from the standard one, which is the focus of this work. Thus, we will not further study the algorithm presented in [7].

As we mention, the idea of proceeding by row is not new. Here, we use the straightforward idea of row pivots to clarify the duality between clear and compress optimisations in the computation of persistent homology and cohomology. In [2, 5, 12], it is shown that, for Vietoris-Rips complexes, the column reduction, coupled with the clear optimisation and applied to the coboundary matrix, provides a significant speed-up in the computation of the barcode. This improvement is not mirrored when the same reduction is coupled with the compress optimisation and applied to the boundary matrix [2, 3, 12]. Here, we show that the reason for this asymmetry is that the second procedure is not the true dual of the first one: to obtain the dual, and thus the same number of operations, it is necessary to reduce the boundary matrix via row operations instead of via column operations.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

2 Preliminaries

Throughout the work, the symbol $K = \{\sigma_1, \ldots, \sigma_m\}$ denote a simplicial complex of dimension d, such that for each $i \leq m$, $K_i \coloneqq \{\sigma_1, \ldots, \sigma_i\}$ is again a simplicial complex. The chain $\emptyset = K_0 \subset K_1 \subset \cdots \subset K_m = K$, denoted by the symbol FK throughout the work, is called a **filtration** of K.

Given a simplex σ of dimension h, its **boundary** is the set of the faces of σ in dimension $h-1$. If $h = 0$, the boundary is empty. Otherwise, it consists of exactly $h+1$ simplices. The **boundary matrix** ∂ of FK, is a $(m \times m)$-matrix with coefficients in $\mathbb{F}_2$ where the j-th column represents the boundary of σ_j, i.e. $\partial[i,j] = 1$ if and only if σ_i is in the boundary of σ_j. Note that, since σ_i is in the complex before σ_j is added, ∂ is an upper-triangular matrix.

For $0 \leq i \leq m$ and $0 \leq p \leq d$, we denote the p-th homology of K_i over the field $\mathbb{F}_2$ by $H_p(K_i)$. The inclusions $K_i \hookrightarrow K_j$, for all $i \leq j$, induce maps on the homology vector spaces $H_p(K_i) \to H_p(K_j)$ for all $0 \leq p \leq d$. This collection of vector spaces and maps forms a diagram called a **persistence module** (Fig. 1(a)) [8].

$$
\begin{array}{lccccccccc}
(a) & 0 & \longrightarrow & V_1 & \longrightarrow & V_2 & \longrightarrow & \cdots & \longrightarrow & V_{m-1} \longrightarrow V_m \\
(b) & 0 & \longrightarrow & \mathbb{F}_2 & \xrightarrow{\mathbf{1}} & \mathbb{F}_2 & \xrightarrow{\mathbf{1}} & \cdots & \xrightarrow{\mathbf{1}} & \mathbb{F}_2 \longrightarrow 0 \\
 & \cdots & & a & & a{+}1 & & \cdots & & b \quad b{+}1
\end{array}
$$

Figure 1 (a) A persistence module; (b) An interval module with interval $[a, b]$.

Since we consider only simplicial complexes with finitely many simplices, the vectors spaces are finite-dimensional. In this case, a persistence module admits a nice decomposition into interval modules, which consist of copies of $\mathbb{F}_2$ connected by the identity morphism for the indices inside an interval, and zero otherwise (Fig. 1(b)) [11]. The collection of intervals in the decomposition of a persistence module is called a **barcode** [6, 15], and it is an invariant of isomorphism type. In [13, Sec. VII], the standard algorithm to retrieve the barcode of a filtered simplicial complex is described. This algorithm retrieves the barcode by studying the lowest elements in the columns of the boundary matrix, whose indices form the so-called (column) pivots (see Definition 3.2). Namely, the algorithm performs left-to-right column additions on columns with the same pivot until no two columns share the same pivot.

3 Row vs column pivot pairing

The (column) pivot pairing in a reduced boundary matrix ∂ provides the lifespan intervals of the homological features of a persistence module [13]. Usually, the reduction is performed using only one type of elementary column operation: adding a column to a later column. Here, we prove that also a reduction performed using only one type of elementary row operation, namely adding a row to a previous row, achieves the same pairing (cf. [14]). The reason why other types of elementary row (column) operations are not allowed is that they do not preserve the order of the generators, and thus cannot maintain the pairing.

Let FK be a filtered simplicial complex, as described in Section 2, with boundary matrix ∂. For the i-th row of ∂, let $left(i)$ denote the column index of the leftmost element of such row. If row i is zero, set $left(i) = 0$.

▶ **Definition 3.1.** A matrix R is called **row reduced** if $left(i) \neq left(i')$ for all non-zero rows $i \neq i'$. An index j is called a **row pivot** (of R) if there exists a row i of R such that $j = left(i)$.

We recall the some standard notions from [13]. The symbol $low(j)$ denotes the index row of the lowest element of column j. If column j is trivial, then $low(j) = 0$.

▶ **Definition 3.2.** A matrix C is called **column reduced** if $low(j) \neq low(j')$ for all non-zero columns $j \neq j'$. An index i is called a **column pivot** (of C) if there exists a column j of C such that $j = low(i)$.

Algorithm 1, called **row pivot reduction**, takes as input the boundary matrix ∂ of a filtered simplicial complex FK and reduces it by row operations. This algorithm is one of the possible methods to achieve a row reduced matrix. Indeed, several different row reduced matrices can be obtained by the same boundary matrix ∂. This follows from the fact that, to the right of each row pivot, there can be several non-zero elements that do not affect the row pivots.

Let m be the number of rows of ∂.

Algorithm 1: Row pivot reduction

Input: Boundary matrix ∂
Output: Row reduced boundary matrix ∂
$R = \partial$
for $i = m, \dots, 1$ **do**
  if $left(i) \neq 0$ **then**
    while *there exists* $i' > i$ *with* $left(i') = left(i) \neq 0$ **do**
      add row i' to row i

In matrix notation, Algorithm 1 computes the reduced matrix as $R = W \cdot \partial$, where W is an invertible upper-triangular matrix with $\mathbb{F}_2$-coefficients.

For a matrix D, consider the following value:

$$r_D(i,j) := \operatorname{rk} D_i^j - \operatorname{rk} D_{i+1}^j + \operatorname{rk} D_{i+1}^{j-1} - \operatorname{rk} D_i^{j-1}$$

where D_i^j is the lower left submatrix of D, given by all the rows of D with index $h \geq i$ and all the columns with index $l \leq j$. The Pairing Lemma [13] states that, for a column reduced matrix C of a boundary matrix ∂, $i = low(j)$ in C if and only if $r_\partial(i,j) = 1$. An analogous result holds for row reduced matrices (cf. [14, Lem 2.2]):

▶ **Lemma 3.3** (Row pairing lemma)**.** *Let ∂ be a boundary matrix and R a row reduced matrix of ∂. Then $j = left(i)$ in R if and only if $r_\partial(i,j) = 1$.*

The proof is precisely the same as the Pairing Lemma [13] since the used technique relies on the lower-left submatrices. Moreover, from the Pairing Lemma [13] and the above Lemma 3.3, $j = left(i)$ in a row reduced matrix R of ∂ if and only if $i = low(j)$ in a column reduced matrix C of ∂. In particular, if $j = left(i)$ or $i = low(j)$, the indices (i,j) form a **persistence pair**.

▶ Remark. Since the coboundary matrix is the anti-transpose of the boundary one (i.e. an element in position (i,j) is sent to position $(m+1-j, m+1-i)$), Algorithm 1 performs the standard column reduction on the coboundary matrix. Indeed, for a reduced matrix R, $j = left(i)$ if and only if $i = low(j)$ in its anti-transpose R^{-T}. Thus, Lemma 3.3 provides an alternative proof of the correctness of the standard persistence algorithm in cohomology, result originally showed in [12].

The running time of Algorithm 1 is at most cubic in the number of simplices, as it is for the standard persistence algorithm. We now refine this estimate a little.

▶ **3.4.** *Computational costs of the reduction.* In [13], it is shown that the running time of the inner (i.e. while) loop in the standard persistence algorithm for the column j, representing a h-simplex σ_j and whose column pivot is in row i, is $(h+1)(j-i)^2$. If σ_j is positive, i.e. at the end of the reduction column j is trivial, then the cost is higher: $(h+1)(j-1)^2$. When reducing the coboundary matrix via column operations, as in [2, 12], the running time becomes $c(j-i)^2$, where c is the number of cofaces of the simplex σ_j, and the cost of reducing a positive column is $c(j-1)^2$. When reducing the boundary matrix via row operations, as in Algorithm 1, the running time of the inner loop in Algorithm 1 for the row i, representing a simplex σ_i and whose row pivot is in column j, is $c(j-i)^2$. Note that, if σ_i is negative, i.e. row i becomes zero at the end of the reduction, then the cost is $c(m-j)^2$, where m is the number of rows. Thus, using row operations, the negative rows are the more expensive to reduce, dually to what happens when reducing by columns.

4 Clear and compress

We now recall two standard runtime optimisations from [3, 9], and show their duality using row and column reductions. Similar observations can be found in [7].

A simplex in the filtered simplicial complex FK is called **positive** if it causes the birth of a homological class, and **negative** if it causes the death of a homological class. By extension, columns and rows in ∂ are called positive (resp. negative) if the corresponding simplices are positive (negative).

▶ **4.1.** *Clear.* The **clear** optimisation is based on the fact that if a row of index j is positive, the j-th column of ∂ cannot be negative. As was already observed in [3, 9], this optimisation is particularly effective when performed on the boundary matrices in decreasing degrees, or, as shown in [5], when applied to the coboundary matrices in increasing degrees. Since the clear avoids reducing columns that are already known not to contain pivots, it is quite helpful in the persistent algorithms up-to-date. However, it is not so useful when reducing by rows, since it shrinks by one the length of each row, but does not avoid any reduction.

▶ **4.2.** *Compress.* The **compress** optimisation hinges on the fact that if a column of index i is negative, the i-th row of ∂ cannot be positive. From Section 3, it follows the real advantages of the compress optimisation are obtained when the matrix reduction is performed using row operations. Indeed, in this case, it avoids a costly loop whose results is already known, while, in accordance with previous results [5], it is quite inefficient when applied using column operations because it only shortens the columns by one element. Performed using the row reduction, the compress is particularly effective when applied to the boundary matrices in increasing degrees.

Finally, for what we showed, in the computation of the barcode of a filtered simplicial complex the number of rows that need to be reduced in the boundary matrix using the compress optimisation is the same as the number of columns that have to be processed in the coboundary matrix when exploiting the clear optimisation. In the case of acyclic complexes, as done in [2, 7], we can be more precise and show that this number is

$$\sum_{h=0}^{d+1} \binom{v-1}{h}$$

where d is the maximal dimension of the acyclic filtered simplicial complex and v the number of vertices. It follows that any algorithm that reduces the coboundary matrix using column operations and the clear can be described as reducing the boundary matrix using row operations and the compress.

Acknowledgments. The author was supported by FAR2019-UniMORE and by the Austrian Science Fund (FWF) grant number P 29984-N35. The author thanks Claudia Landi, Michael Kerber, Wojciech Chachólski, and Håvard B. Bjerkevik for useful discussions and feedback.

References

1 Ulrich Bauer. *Ripser: a lean C++ code for the computation of Vietoris–Rips persistence barcodes*. https://ripser.org, 2015-2020.

2 Ulrich Bauer. Ripser: efficient computation of Vietoris-Rips persistence barcodes, 2019. arXiv:1908.02518.

3 Ulrich Bauer, Michael Kerber, and Jan Reininghaus. Clear and compress: Computing persistent homology in chunks. *Mathematics and Visualization*, 03 2013. doi:10.1007/978-3-319-04099-8_7.

4 Ulrich Bauer, Michael Kerber, Jan Reininghaus, and Hubert Wagner. *PHAT - Persistent Homology Algorithms Toolbox*. https://bitbucket.org/phat-code/phat/src/master/.

5 Ulrich Bauer, Michael Kerber, Jan Reininghaus, and Hubert Wagner. Phat—persistent homology algorithms toolbox. *J. Symbolic Comput.*, 78:76–90, 2017. doi:10.1016/j.jsc.2016.03.008.

6 Gunnar Carlsson, Afra Zomorodian, Anne Collins, and Leonidas Guibas. Persistence Barcodes for Shapes. In Roberto Scopigno and Denis Zorin, editors, *Symposium on Geometry Processing*. The Eurographics Association, 2004. doi:10.2312/SGP/SGP04/127-138.

7 Wojciech Chachólski, Barbara Giunti, Alvin Jin, and Claudia Landi. Algorithmic decomposition of filtered chain complexes, 2020. arXiv:2012.01033.

8 Frédéric Chazal, Vin de Silva, Marc Glisse, and Steve Oudot. *The structure and stability of persistence modules*. SpringerBriefs in Mathematics. Springer, [Cham], 2016. doi:10.1007/978-3-319-42545-0.

9 Chao Chen and Michael Kerber. Persistent homology computation with a twist. In *Proceedings 27th European Workshop on Computational Geometry*, 2011.

10 David Cohen-Steiner, Herbert Edelsbrunner, and Dmitriy Morozov. Vines and vineyards by updating persistence in linear time. In *Computational geometry (SCG'06)*, pages 119–126. ACM, New York, 2006. doi:10.1145/1137856.1137877.

11 William Crawley-Boevey. Decomposition of pointwise finite-dimensional persistence modules. *J. Algebra Appl.*, 14(5):1550066, 8, 2015. doi:10.1142/S0219498815500668.

12 Vin de Silva, Dmitriy Morozov, and Mikael Vejdemo-Johansson. Dualities in persistent (co)homology. *Inverse Problems*, 27(12):124003, 17, 2011. doi:10.1088/0266-5611/27/12/124003.

13 Herbert Edelsbrunner and John L. Harer. *Computational topology*. American Mathematical Society, Providence, RI, 2010. An introduction. doi:10.1090/mbk/069.

14 Herbert Edelsbrunner and Katharina Ölsböck. Tri-partitions and bases of an ordered complex. *Discrete & Computational Geometry*, 64(3):759–775, Oct 2020. doi:10.1007/s00454-020-00188-x.

15 Robert Ghrist. Barcodes: the persistent topology of data. *Bull. Amer. Math. Soc. (N.S.)*, 45(1):61–75, 2008. doi:10.1090/S0273-0979-07-01191-3.

16 Database of real-world applications of TDA. *https://www.zotero.org/groups/2425412/tda-applications*. 2020.

17 Nina Otter, Mason Porter, Ulrike Tillmann, Peter Grindrod, and Heather Harrington. A roadmap for the computation of persistent homology. *EPJ Data Science*, 6, 06 2015.

Improved Approximation Bounds for Half-Guarding Monotone Polygons

Hannah Miller Hillberg[1], Erik Krohn[1], and Alex Pahlow[1]

1 **University of Wisconsin - Oshkosh**
hillbergh,krohne,pahloa45@uwosh.edu

Abstract

We consider a variant of the art gallery problem where all guards are limited to seeing to the right inside a monotone polygon. We provide a polynomial-time approximation for point guarding the entire monotone polygon. We improve the best known approximation of 40 from [11], to 8.

1 Introduction

An instance of the *art gallery problem* takes as input a simple polygon P. A polygon P is defined by a set of points $V = \{v_1, v_2, \ldots, v_n\}$. There are edges connecting (v_i, v_{i+1}) where $i = 1, 2, \ldots, n-1$. There is also an edge connecting (v_1, v_n). If these edges do not intersect other than at adjacent points in V (or at v_1 and v_n), then P is called a simple polygon. For any two points $p, q \in P$, we say that p sees q if the line segment $\overline{pq}$ does not go outside of P. The art gallery problem seeks to find a guarding set of points $G \subseteq P$ such that every point $p \in P$ is seen by a point in G. In the point guarding problem, guards can be placed anywhere inside of P. In the vertex guarding problem, guards are only allowed to be placed at points in V. The optimization problem is defined as finding the smallest such G in each case.

1.1 Previous Work

There are many results about guarding art galleries. Several results related to hardness and approximations can be found in [2, 6, 7, 8, 9, 14]. Whether a polynomial time constant factor approximation algorithm can be obtained for vertex guarding a simple polygon is a longstanding and well-known open problem, although a claim for one was made in [4].

Additional Polygon Structure. Due to the inherent difficulty in fully understanding the art gallery problem for simple polygons, there has been some work done guarding polygons with additional structure, see [3, 5, 11, 13] for example. In this paper we consider *monotone polygons*, described below.

α-Floodlights. Motivated by the fact that many cameras and other sensors often cannot sense in 360°, previous works have considered the problem when guards have a fixed sensing angle α for some $0 < \alpha \leq 360$. This problem is often referred to as the *α-floodlight problem*. More specifically, a *half-guard* is defined as a 180°-floodlight that sees only in one direction. This subset of the floodlight problem is motivated by considering the polygon to be a time sequence as one works from left to right. The leftmost point is time 0 and time increases as one sweeps from left to right. A guard cannot see back in time, therefore, seeing back to the left does not make sense. This variant attempts to model time in the problem by ensuring that a region is guarded in a specific time range from the time it is placed until some time in the future. A constant factor approximation for half-guarding was first shown in [11] and NP-hardness with vertex guards was shown in [10]. Some of the work on the α-floodlight problem has involved proving necessary and sufficient bounds on the number

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

of α-floodlights required to guard (or illuminate) an n vertex simple polygon P, where floodlights are anchored at vertices in P and no vertex is assigned more than one floodlight, see for example [15, 16]. Computing a minimum cardinality set of α-floodlights to illuminate a simple polygon P is APX-hard for both point guarding and vertex guarding [1].

1.2 Our Contribution

In this paper, we improve the approximation bounds for half-guarding monotone polygons. A simple polygon P is *x-monotone* (or simply *monotone*) if any vertical line intersects the boundary of P in at most two points. A *half-guard* can see only to the right, so we redefine *sees* as: a point p sees a point q if the line segment $\overline{pq}$ does not go outside of P and $p.x \leq q.x$, where $p.x$ denotes the x-coordinate of a point p. In a monotone polygon, let l and r denote the leftmost and rightmost point of P respectively. Consider the "top half" of the boundary of P by walking along the boundary clockwise from l to r. We call this the *ceiling* of P. We obtain the *floor* of P by walking counterclockwise along the boundary from l to r.

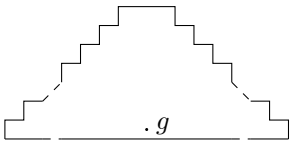


Figure 1 A regular guard can see this entire monotone polygon, but needs $\Omega(n)$ half-guards.

Krohn and Nilsson [13] give a constant factor approximation for monotone polygons using guards that can see 360°. There are monotone polygons P that can be completely guarded with one guard that require $\Omega(n)$ half-guards considered in this paper, see Figure 1. Due to the restricted nature of half-guards, new observations are needed to obtain the approximation given in this paper. A 40-approximation for this problem was presented in [11]. The algorithm in [11] places guards in 5 steps: guard the ceiling vertices, then the floor vertices, then the entire ceiling boundary, then the entire floor boundary, and finally any missing portions of the interior. We propose a modified algorithm that requires only 3 steps: guarding the entire ceiling, then the entire floor, and lastly any missing portions of the interior. By modifying the algorithm and providing improved analysis, we obtain an 8-approximation.

The remainder of the paper is organized as follows. Section 2 gives an algorithm for point guarding a monotone polygon using half-guards where we wish to guard the boundary of the polygon. Section 3 provides a sketch of the 8-approximation.

2 Guarding the Boundary

In this section, we give an algorithm for guarding the boundary of a monotone polygon P with half-guards that see to the right. We first give a 2-approximation algorithm for guarding the entire ceiling. A symmetric algorithm works for guarding the entire floor giving us a 4-approximation for guarding the entire boundary of the polygon.

Before we describe the algorithm, we provide some preliminary definitions. A vertical line that goes through a point p is denoted l_p. Given two points p, q in P such that $p.x < q.x$, we use (p, q) to denote the points s such that $p.x < s.x < q.x$. Similarly, we use $(p, q]$ to denote points s such that $p.x < s.x \leq q.x$.

2.1 Ceiling Guard Algorithm

We first give a high level overview of the algorithm for guarding the entire boundary of the ceiling. Any feasible solution must place a guard at the leftmost vertex where the ceiling and floor come together (or this vertex would not be seen). We begin by placing a guard here. We iteratively place guards from left to right. When placing the next guard, we let S denote the guards the algorithm has already placed, and we let p denote the leftmost point on the ceiling that is not seen by any guard in S. Note that p may be a ceiling vertex or any point on the ceiling. The next guard g that is placed will lie somewhere on the line l_p. We initially place g at the intersection of l_p and the floor, and we slide g upwards vertically along l_p. The algorithm locks in a final position for the guard g by sliding it upwards along l_p until moving it any higher will cause g to no longer see some unseen portion on the ceiling; let r be the first such point. See, for example, Figure 2. In this figure, when g is initially placed on the floor, it does not see r, but as we slide g up the line l_p, r becomes a new point that g can see. If we slide g up any higher than as depicted in the figure, then g would no longer see r, and therefore we lock g in that position. We then add g to S, and we repeat this procedure until the entire ceiling is guarded. The ceiling guarding algorithm is shown in Algorithm 1.

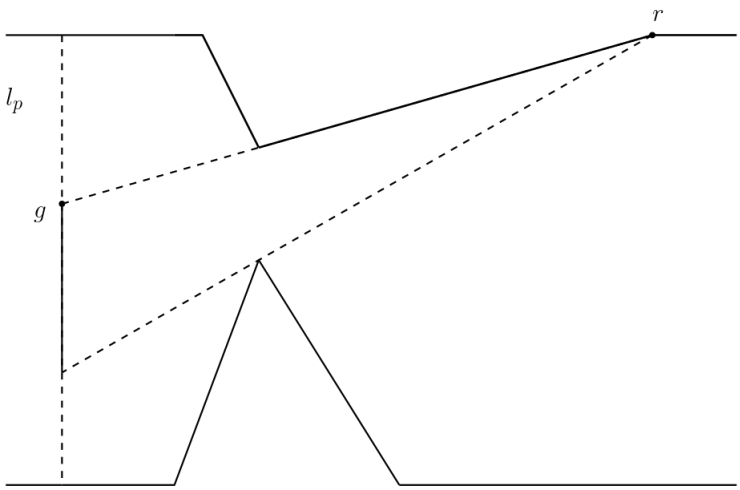


Figure 2 A guard g slides up l_p and sees a point r. If g goes any higher, it will stop seeing r.

Algorithm 1 clearly returns a set of guards that sees the entire ceiling. All steps, except the sliding step, can be trivially done in polynomial time. The analysis of [11] uses a similar sliding step but only considers guarding a polynomial number of vertices on the ceiling or floor. When considering an infinite number of points on the ceiling, it is not immediately clear that the sliding can be done in polynomial time since each time a guard moves an ϵ amount upwards, it will see a different part of the boundary. We use the following lemma to help bound the number of locations a guard g must consider on l_p.

▶ **Lemma 1.** *Consider a guard g and a point on the floor f such that $g.x < f.x$. If g sees f, then the floor cannot block g from seeing any ceiling point in (g, f).*

Proof. By assumption, g sees f. Consider a point on the ceiling p such that $g.x < p.x < f.x$. If g is being blocked from seeing p because of a floor vertex, then g cannot see any point to the right of p, see Figure 3. Therefore, g cannot see f and we have a contradiction. ◀

Algorithm 1 Ceiling Guard

```
1: procedure CEILING GUARD(monotone polygon P)
2:     S ← {g} such that g is placed at the leftmost point l.
3:     while there is a point on the ceiling that is not seen by a guard in S do
4:         Let p be the leftmost ceiling point that is currently unseen by any guards in S.
           Place a guard g where l_p intersects the floor and slide g up. As g is being slid up, let
           r be the first point on the ceiling that g would stop seeing if g moved any further up.
           Place g at the highest location on l_p such that g sees r.
5:         S ← S ∪ {g}.
6:     end while
7:     return S
8: end procedure
```

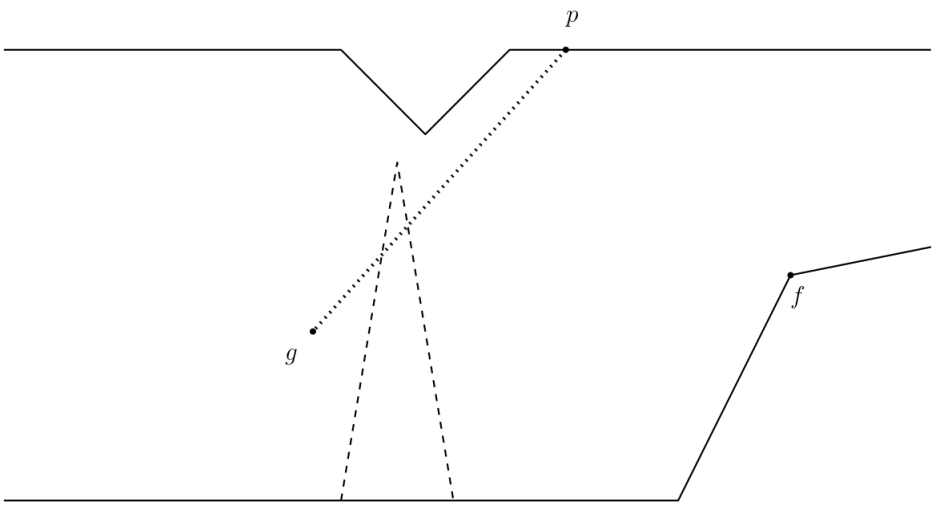


Figure 3 If the floor blocks g from a ceiling point p, then g will not see any floor point f where $g.x < p.x < f.x$. More generally, g will not see any point f where $g.x < p.x < f.x$.

We now give a sketch of a proof for why there are at most $O(n^3)$ potential guard locations on l_p that must be considered. The approximation analysis sketched later in Section 3 relies on the fact that g is as high as possible on l_p. If g is moved higher, then there is a point r on the ceiling that would not be seen. In simple polygons, any point that is seen by a point on l_p must by seen by a contiguous line segment of l_p, see Figure 2. There are several cases to consider on whether or not g is moved upwards and how far it needs to move.

In the cases below, whenever the algorithm is sliding a guard, one only needs to consider the following locations on l_p: (A) Rays shot from a vertex through another vertex until it hits l_p. There are $O(n^2)$ potential locations on l_p. (B) Shoot a ray from previously placed guards $g' \in S$ through every vertex. Let $C(g')$ be the set of all of the points on the ceiling that these rays hit. Let $C = \bigcup_{g' \in S} C(g')$. Shoot a ray from all $c \in C$ through all vertices until the ray hits l_p. There are at most n guards, therefore $|C(g')| \leq n$. For each guard, there are at most $O(n^2)$ potential locations on l_p to consider for a total of $O(n^3)$ locations.

Case 1: If there exists some vertex v_i such that g sees vertex v_i, but does not see vertex v_{i+1} because v_i is blocking g from seeing v_{i+1}, see Figure 4. If there are multiple v_i candidates, we choose the v_i that is leftmost. Shoot a ray from g through v_i and let r be the point on

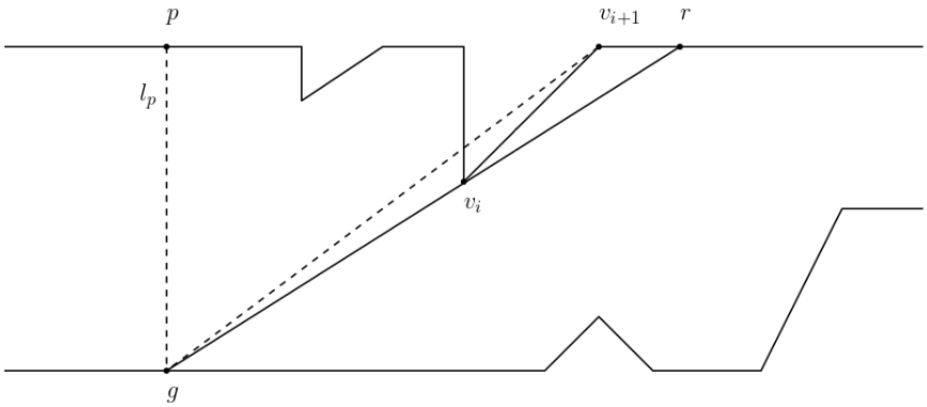


Figure 4 r is seen by g and g is the leftmost guard that sees r.

the boundary that is hit.

Case 1a: If r is on the ceiling, then no guard to the left of g is able to see r. If g is slid up, then g would no longer see r. In other words, g is the leftmost guard that sees r. In this case, we place g on the floor.

Case 1b: If r is on the floor, then moving g upwards will not see any more ceiling points to the right of r because v_i is blocking g from seeing them, see Figure 5. By Lemma 1, moving g upwards will not result in seeing any more ceiling points since the floor cannot be blocking g from any point on the ceiling to the left of r. However, the approximation analysis relies on r being as high as possible on l_p. Therefore, we slide g upwards until it would have stopped seeing some previously unseen point r' on the ceiling and set $r = r'$. If no previously unseen ceiling point $r' \in [p, v_i]$ exists, then it must be the case that all points on l_p see all of the ceiling points of $[p, v_i]$. In this case, we place g on the ceiling at point p and set $r = v_i$.

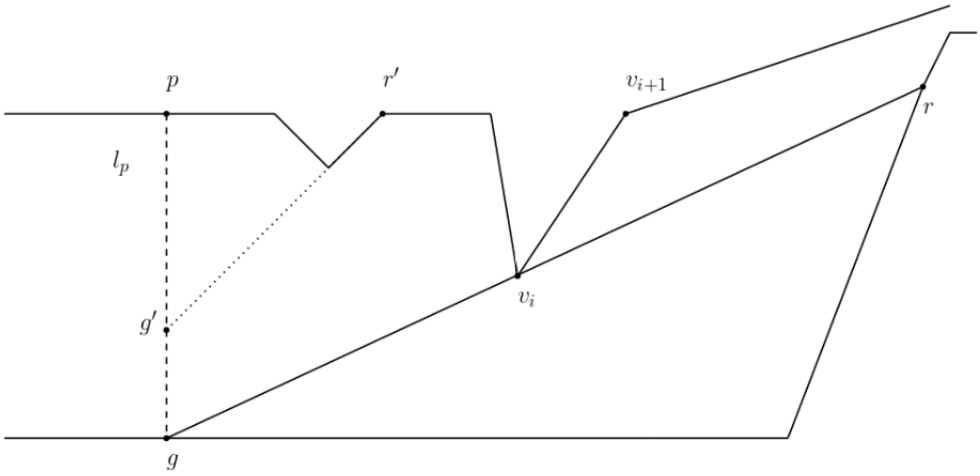


Figure 5 Case 1b where g cannot see v_{i+1} because v_i is blocking it and r is on the floor.

Case 2: If, for all v_i that are seen by g, g is not blocked from seeing v_{i+1} by v_i, we slide g upwards. If none of Case 1 happens while sliding upwards, then the algorithm stops whenever g would stop seeing a previously unseen point on the ceiling. In Figure 6, a portion of the edge $e = [v_{i+1}, v_{i+2}]$, namely $[v_{i+1}, x]$, was seen by previously placed guards in the algorithm.

It is possible that previously placed guards saw a portion of e between $(x, v_{i+2}]$. However, we consider the rightmost point x such that all of $[v_{i+1}, x]$ is seen. Let r be the point directly to the right of x. If g is pushed up too far (e.g. g_3) in order to see v_{i+2}, it ends up missing a portion of e, namely r. One must slide g between the floor point of l_p and g_3 to ensure the algorithm places guards that see all of e. In this case, we wish to place a guard on l_p as high as possible such that g sees r. Let g_2 be the highest point on l_p that sees r. Placing g above g_2 will cause it to miss seeing point r. Placing g lower than g_2 might allow a guard in the optimal solution to see r and also see more of e than g saw. Therefore, by using the set of guard locations as defined in (B) above, g would be placed at g_2. The approximation relies on the fact that g is as high as possible on l_p such that it still sees r.

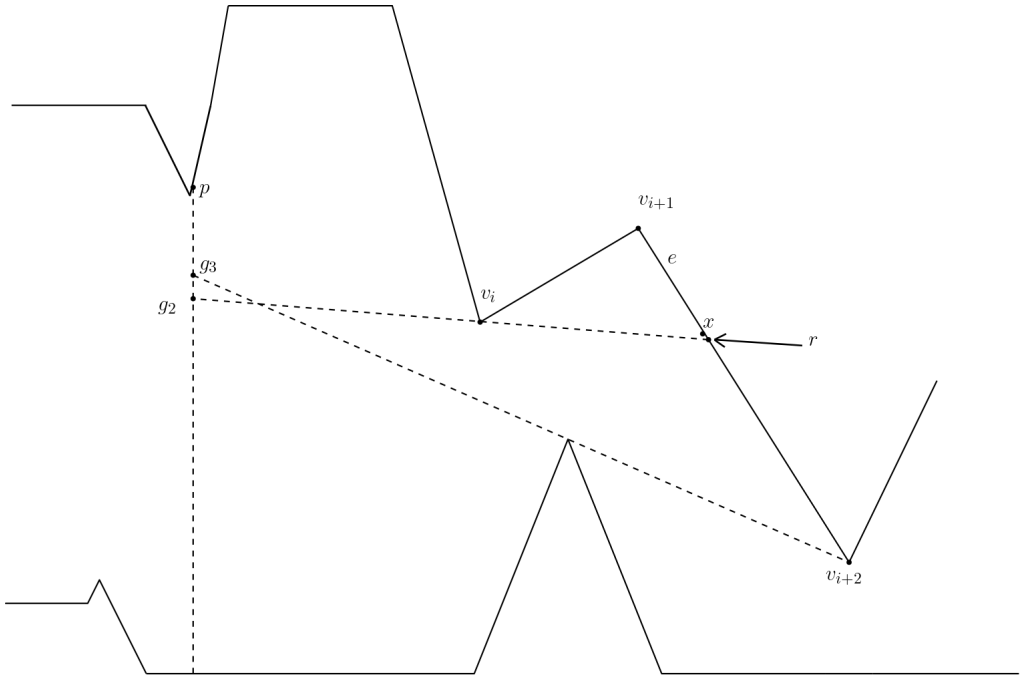


Figure 6 No point on l_p sees all of (x, v_{i+2}). Algorithm places a guard at g_2 to ensure r is seen.

3 Sketch of Approximation

We will now sketch out the proof of why Algorithm 1 will place no more than 2 times the number of guards in the optimal solution. An optimal solution $\mathcal{O}$ is a minimum cardinality guard set such that for any point p on the ceiling of P, there exists some $g \in \mathcal{O}$ that sees p. The argument will be a charging argument; every guard placed will be charged to a guard in $\mathcal{O}$ in a manner such that each guard in $\mathcal{O}$ will be charged at most twice. The approximation argument similar to [11] is sketched here. Consider two consecutive guards $g_i, g_{i+1} \in S$ returned by Algorithm 1:

Case 1: If an optimal guard is in $(g_i, g_{i+1}]$, we charge g_{i+1} to that optimal guard.

Case 2: If there is no optimal guard in $(g_i, g_{i+1}]$, then consider the point on the ceiling directly above g_{i+1}, call this point p. g_{i+1} was placed on l_p because no previously placed guards saw p. Consider an optimal guard o that sees p. $o.x \leq g_{i+1}.x$ and even more so, $o.x \leq g_i.x$ since, by assumption, there is no optimal guard in $(g_i, g_{i+1}]$. Since g_i did not see p, it must be the case that the $\overline{op}$ line goes above g_i and the floor blocks g_i from seeing p, see

Figure 7. The $\overline{op}$ line cannot be below g_i. If it were, then g_i would have had the opportunity to see p as g_i was moving upwards. This is not possible because by assumption, p is not seen by any previously placed guard and g_i would not have gone so far upwards as to stop seeing a previously unseen ceiling point.

Since g_i is blocked from seeing p by the floor, g_i cannot see any ceiling points to the right of p. The reason that g_i stopped moving upwards is because it saw some point r on the ceiling that no previous guard saw. Since g_i cannot see to the right of p, r must be to the left of p. By assumption, the optimal guard o' that sees r must be to the left of g_i. If g_i were to have moved any higher up, it would have missed r. Any guard that sees r must be "below" the $\overline{g_i r}$ line, see Figure 7. Any point on the ceiling that o' sees to the right of r, g_i will also see (Lemma 1, [11]). Therefore, g_i dominates o' with respect to the ceiling to the right of r. We charge g_{i+1} to o' and o' cannot be charged again.

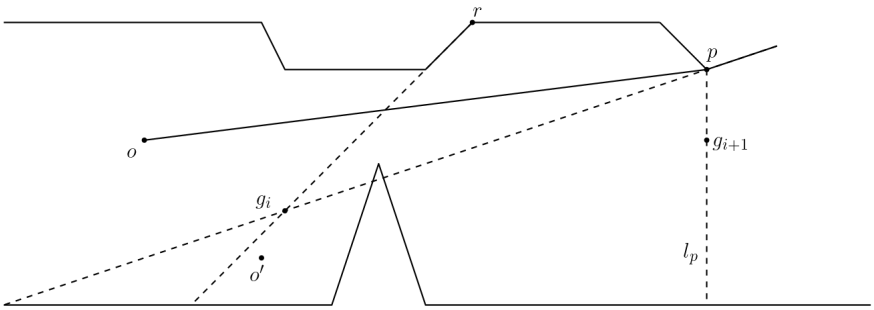


Figure 7 If no optimal guard exists in $(g_i, g_{i+1}]$, then o' must exist to see r such that $o'.x \leq g_i.x$.

The entire ceiling can be guarded with at most $2 \cdot |\mathcal{O}|$ guards. A similar algorithm is applied to the floor to give at most $4 \cdot |\mathcal{O}|$ guards to guard the entire boundary. Finally, even though the entire boundary is guarded, it is possible that a portion of the interior is unseen, see Figure 8. Let us assume that a guardset $G = \{g_1, g_2, \ldots g_k\}$ guards the entire boundary of a monotone polygon such that for all i, $g_i < g_{i+1}$. In [12], they prove that between any consecutive guards of G, a region can exist that is unseen by any of the guards in G. However, they prove that the region is convex and can be guarded with 1 additional guard. If $|G| = k$, then there are at most $k - 1$ guards that need to be added to guard these unseen interior regions. This doubles the approximation to give us the following theorem.

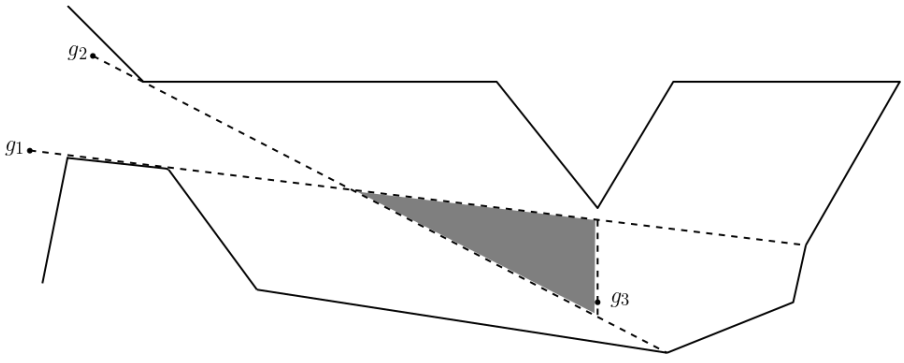


Figure 8 g_1, g_2 and g_3 see all of the boundary. The shaded region is unguarded.

▶ **Theorem 2.** *There is a polynomial-time* 8*-approximation algorithm for point guarding a monotone polygon with half-guards.*

References

1. Ahmed Abdelkader, Ahmed Saeed, Khaled A. Harras, and Amr Mohamed. The inapproximability of illuminating polygons by α-floodlights. In *CCCG*, pages 287–295, 2015.
2. Alok Aggarwal. *The art gallery theorem: its variations, applications and algorithmic aspects*. PhD thesis, The Johns Hopkins University, 1984.
3. Pritam Bhattacharya, Subir Ghosh, and Bodhayan Roy. Approximability of guarding weak visibility polygons. *Discrete Applied Mathematics*, 228, 02 2017. doi:10.1016/j.dam.2016.12.015.
4. Pritam Bhattacharya, Subir Kumar Ghosh, and Sudebkumar Prasant Pal. Constant approximation algorithms for guarding simple polygons using vertex guards. *CoRR*, abs/1712.05492, 2017. URL: http://arxiv.org/abs/1712.05492, arXiv:1712.05492.
5. Pritam Bhattacharya, Subir Kumar Ghosh, and Bodhayan Roy. Vertex guarding in weak visibility polygons. In Sumit Ganguly and Ramesh Krishnamurti, editors, *Algorithms and Discrete Applied Mathematics*, pages 45–57, Cham, 2015. Springer International Publishing.
6. Björn Brodén, Mikael Hammar, and Bengt J. Nilsson. Guarding lines and 2-link polygons is APX-hard. In *CCCG*, pages 45–48, 2001.
7. Alon Efrat and Sariel Har-Peled. Guarding galleries and terrains. *Information Processing Letters*, 100(6):238–245, 2006.
8. Stephan Eidenbenz. Inapproximability results for guarding polygons without holes. In *ISAAC*, pages 427–436, 1998.
9. Subir Kumar Ghosh. On recognizing and characterizing visibility graphs of simple polygons. *Discrete & Computational Geometry*, 17(2):143–162, 1997.
10. Matt Gibson, Erik Krohn, and Matt Rayford. Guarding monotone polygons with vertex half-guards is np-hard. *European Workshop on Computational Geometry*, 2018. URL: https://conference.imp.fu-berlin.de/eurocg18/program.
11. Matt Gibson, Erik Krohn, and Matthew Rayford. Guarding monotone polygons with half-guards. In Joachim Gudmundsson and Michiel H. M. Smid, editors, *Proceedings of the 29th Canadian Conference on Computational Geometry, CCCG 2017, July 26-28, 2017, Carleton University, Ottawa, Ontario, Canada*, pages 168–173, 2017.
12. Erik Krohn and Bengt J. Nilsson. The complexity of guarding monotone polygons. In *CCCG*, pages 167–172, 2012.
13. Erik Krohn and Bengt J. Nilsson. Approximate guarding of monotone and rectilinear polygons. *Algorithmica*, 66(3):564–594, 2013.
14. D. T. Lee and A. K. Lin. Computational complexity of art gallery problems. *IEEE Trans. Inform. Theory*, 32(2):276–282, March 1986.
15. Bettina Speckmann and Csaba D. Tóth. Allocating vertex π-guards in simple polygons via pseudo-triangulations. *Discrete & Computational Geometry*, 33(2):345–364, 2005.
16. Csaba D. Tóth. Art galleries with guards of uniform range of vision. *Computational Geometry*, 21(3):185 – 192, 2002.

Mapping Multiple Regions to the Grid with Bounded Hausdorff Distance

Ivor van der Hoog[1], Mees van de Kerkhof[1], Marc van Kreveld[1], Maarten Löffler[1], Frank Staals[1], Jérôme Urhausen[1], and Jordi L. Vermeulen[1]

1 **Utrecht University, the Netherlands**
{I.D.vanderHoog, M.A.vandeKerkhof, M.J.vanKreveld, M.Loffler, F.Staals, J.E.Urhausen, J.L.Vermeulen}@uu.nl

Abstract

We study a problem motivated by digital geometry: given a set of disjoint geometric regions, assign each region R_i a set of grid cells P_i, so that P_i is connected, similar to R_i, and does not touch any grid cell assigned to another region. Similarity is measured using the Hausdorff distance. We prove an asymptotically tight bound on the achievable Hausdorff distance for convex input regions in terms of the number of input regions, and prove that there is no upper bound to the Hausdorff distance for three or more general regions.

1 Introduction

Digital geometry is concerned with the proper representation of geometric objects and their relationships using a grid of pixels. This greatly simplifies both representation and many operations, but the downside is that common properties of geometric objects no longer hold. For example, it may be that two digitized lines intersect in multiple connected components. How to digitize a set of geometric objects so that such properties are guaranteed is one objective in digital geometry, referred to as *consistency*. Another objective is the representation of vector objects with bounded error, using subsets of pixels. Here we may assume the unit grid, and measure error in one of multiple ways.

Early results in digital geometry were mostly concerned with consistency and arose in computer vision. For a survey, see Klette and Rosenfeld [13, 14]. The interest from the algorithms community is more recent. Besides consistency, the Hausdorff distance of digital representations is a topic of study. Chun et al. [7] investigate the problem of representing rays originating in the origin as digital rays such that certain properties are satisfied. They show that rays can be represented on the $n \times n$ grid in a consistent manner with Hausdorff distance $O(\log n)$. This bound is tight in the worst case. By ignoring one of the consistency

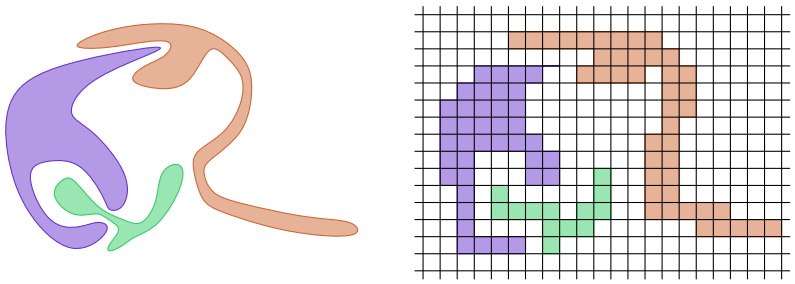


Figure 1 Three disjoint simply-connected regions and a grid representation of them.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

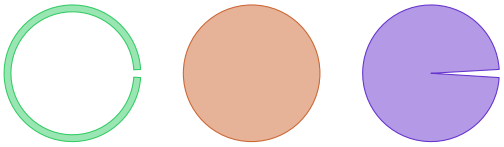


Figure 2 The Hausdorff distance between the green and red regions is large while the Hausdorff distance between their boundaries is small. The inverse is true for the red and purple regions.

conditions, the distance bound improves to $O(1)$. Their research is extended by Christ et al. [5] to line segments (not necessarily starting in the origin), who obtain the logarithmic distance bound in this case as well. A possible extension to curved rays was developed by Chun et al. [6]. Recently, Chiu and Korman [3] extended some results to high dimensional segments. Additionally, Chiu et al. [4] prove that the above mentioned lower bound of $\Omega(\log n)$ does not extend to higher dimensions and proved an $\Omega(\log^{1/(1-d)} n)$ bound instead. Other results with a digital geometry flavor within the algorithms community are those on snap rounding [8, 9, 12], integer hulls [1, 11], and discrete schematization [15].

In a recent paper, Bouts et al. [2] showed that any simple polygon, no matter how detailed, can be represented by a simply-connected set of unit pixels such that the Hausdorff distance to and from the input is bounded by the constant $\frac{3}{2}\sqrt{2}$. They also prove that for the Fréchet distance between the boundaries, a constant distance is not possible. In this paper we extend their results to multiple regions, see Figure 1 for an example.

In Section 2, we show that when we restrict the m input regions to be convex, we can find a grid representation within Hausdorff distance $\Theta(m)$. This bound is tight. Furthermore, we can extend the result from Bouts et al. [2] to two general regions; that is, we can find a representation with constant Hausdorff distance. This is in contrast with our proof in Section 3 that for three or more general regions, there is no universal bound on the Hausdorff distance of a grid representation.

We do not make any assumptions on the resolution of the input. If the minimum distance between any pair of polygons is at least some constant (e.g., $4\sqrt{2}$ is enough), then we can realize a constant Hausdorff bound in all cases by applying the results from Bouts et al. [2] separately on each polygon. We consider the case where no such assumptions are made.

Notation and definitions. We denote by Γ the (infinite) unit grid, whose unit squares are referred to as *pixels*. The *(symmetric) Hausdorff distance* between two sets $A, B \subset \mathbb{R}^2$ is defined as $H(A,B) = \max\{\max_{a\in A}(\min_{b\in B}(|ab|)), \max_{b\in B}(\min_{a\in A}(|ab|))\}$, where $|ab|$ is the distance between the points a and b. Furthermore, we denote by $H'(A,B) = \max\{H(A,B), H(\partial A, \partial B)\}$ the maximum of the (symmetric) Hausdorff distance between the sets themselves and between their boundaries. See Figure 2 for an example where the distinction between $H(\cdot,\cdot)$ and $H'(\cdot,\cdot)$ is important.

Consider a set of m disjoint simply-connected regions $\mathcal{R} = \{R_1, R_2, \ldots R_m\}$ in the plane, which we can imagine having different colors. A choice of pixels for each region can then be seen as a *coloring* of the grid by m colors $c_1, \ldots, c_m$, where pixels may remain uncolored.

In this paper, we show how to assign a simply-connected subset of the pixels $P_i \subset \Gamma$ to each region $R_i \in \mathcal{R}$, such that the result is a set of m disjoint simply-connected regions. Two such *grid polygons* are disjoint if they do not meet in any edge or vertex of the grid. A grid polygon is connected if its pixels are connected by edge adjacency. A grid polygon is simply-connected if it is connected and its complement is also connected by edge adjacency. Hence, we do not allow vertex adjacency at all as it is ambiguous. In this paper a grid

polygon is by default simply-connected. We call the set $P_1, P_2, \ldots, P_m$ of such grid polygons a *valid assignment*, see for example Figure 1. We are interested in finding for any set of regions $\mathcal{R}$ a valid assignment where for each region $R_i \in \mathcal{R}$, its corresponding grid polygon P_i has a (symmetric) Hausdorff distance to R_i of at most h and their boundaries are also within (symmetric) Hausdorff distance h.

2 Input regions are convex regions

When $\mathcal{R}$ is a set of convex regions, we can easily show that a coloring has a Hausdorff distance of $\Omega(m)$ in the worst case: we place m horizontal line segments of length $\Omega(m)$ that all pass through the same pixels. Then $\mathcal{P}$ must have its elements on disjoint lines of pixels, giving Hausdorff distance at least $\Omega(m)$ for the outer regions. Each P_i must extend sufficiently far left and right. Since each P_i is connected, all P_i will intersect a common vertical line. The topmost or bottommost intersection with this line belongs to a grid polygon with Hausdorff distance $\Omega(m)$.

We will describe an algorithm that, given a set of convex regions $\mathcal{R}$, gives a set of disjoint grid polygons $\mathcal{P}$ that is a valid assignment and such that for all i, $H'(R_i, P_i) = O(m)$.

▶ **Observation 2.1.** *Let $R_1, R_2 \in \mathcal{R}$ be two disjoint convex regions, and let ℓ be a horizontal line that intersects R_1 left of R_2. Then any horizontal line intersecting both R_1 and R_2 intersects R_1 left of R_2. Analogously, all vertical lines that intersect both R_1 and R_2 do so in the same above-below order.*

Observation 2.1 allows us to define two partial orders $\preceq_x$ and $\preceq_y$ on $\mathcal{R}$: $R_i \preceq_x R_j$ if and only if there is a horizontal line intersecting both regions and R_i intersects the line left of R_j; since the regions are convex we get a partial order [10]. We extend this partial order to a linear order $X_{\mathcal{R}} : \mathcal{R} \to [1, m]$ in any manner. A linear order $Y_{\mathcal{R}} : \mathcal{R} \to [1, m]$ is defined symmetrically.

Given $X_{\mathcal{R}}$ and $Y_{\mathcal{R}}$, we assign a coloring as illustrated in Figure 3. Let Γ_k be a coarsening of the grid Γ whose cells have $k \times k$ pixels with $k = 2m$. The factor 2 ensures that adjacent polygons do not touch. These cells are also called *superpixels*. In the following, we ensure that if the region R_i intersects a superpixel, this superpixel or an adjacent one contains a pixel of P_i and vice versa. Thus, we get a bound of $O(k)$ on the Hausdorff distance $H(R_i, P_i)$. For any superpixel $S \in \Gamma_k$, we denote by $S[x, y]$ the pixel that is the $(2x)^{\text{th}}$ from the left and $(2y)^{\text{th}}$ from the bottom within S. The horizontal and vertical lies induced by Γ_k are called *major lines*. Each region R_i that intersects at most one major horizontal line and at most one major vertical major line is a *small region*. Each region R_i that intersects at least two major horizontal lines or at least two major vertical lines is a *large region*. Our assignment of regions to pixels works as follows:

1. For each small region R_i we choose one superpixel S containing a point of R_i and color the pixel $p(S, R_i) := S[X_{\mathcal{R}}(R_i), Y_{\mathcal{R}}(R_i)]$ with c_i; this single pixel will be P_i.
2. For each superpixel S and each large region R_i intersecting S that also intersects the two major horizontal lines incident to S, or the two major vertical lines incident to S, we color $p(S, R_i) = S[X_{\mathcal{R}}(R_i), Y_{\mathcal{R}}(R_i)]$ with c_i. We use full lines, not the edges of S here.
3. For any two pixels that are colored with c_i in edge-adjacent superpixels (R_i must be large), we color all pixels in the row or column between them with c_i as well.
4. For any four superpixels that share a common vertex, if they each contain a pixel colored with c_i in Step 2, we color all pixels in the square between these pixels with c_i as well.

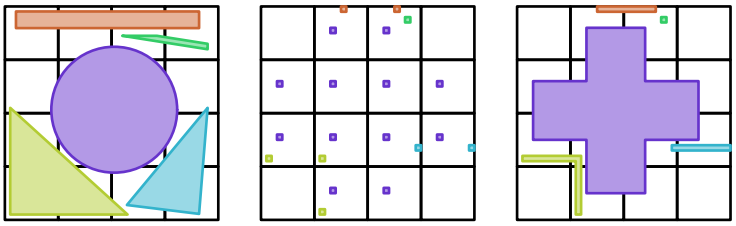


Figure 3 The coloring algorithm for convex regions. (a) shows the input of five convex regions, overlaid onto a superpixel grid with $k = 10$. (b) shows the pixels colored in Steps 1 and 2 of the algorithm. (c) shows the final coloring obtained after Steps 3 and 4.

The set $\mathcal{P}$ of polygons induced by this grid coloring is a valid assignment, that is, the polygons are connected and do not intersect or touch.

Overall, if a region R_i intersects a superpixel S, then P_i has a pixel in S or in any of the 8 adjacent superpixels. Conversely, if P_i has a pixel in a superpixel S, we know that R_i intersects S. This gives a bound on the Hausdorff distance between the regions and the grid polygons. For the boundaries, note that if R_i contains a superpixel S and all four edge-adjacent superpixels, then P_i contains S. Furthermore, if P_i contains a superpixel S, then R_i also contains S. Together this gives a bound on the Hausdorff distance between the boundaries. Since superpixels have size $\Theta(m)$, the Hausdorff distance between R_i and P_i and between their boundaries is at most $O(m)$. We thus obtain the following result.

▶ **Theorem 2.2.** *If $\mathcal{R}$ consists of m convex regions, a valid assignment to regions exists such that for each region $R_i \in \mathcal{R}$ and grid polygon P_i, we have $H'(R_i, P_i) = O(m)$. Furthermore, there exists a set $\mathcal{R}$ of m convex regions such that for every valid assignment, there exists some $1 \leq i \leq m$ with $H(R_i, P_i) = \Omega(m)$.*

3 Two or three general regions

We extend the result from Bouts et al. [2] to two general regions using the result from van Goethem et al. [16], that is, we can find two grid polygons with constant Hausdorff distance to two input regions. Below, as a stark contrast, we show that the Hausdorff distance between an input of at least three general regions and any corresponding grid polygons is unbounded. Formally, for a given integer $h > 0$, we show a construction of regions $\mathcal{R} = \{R, B, G\}$ for which there is no valid assignment of corresponding grid polygons with Hausdorff distance smaller than h.

We only sketch the main idea here. We construct regions $\mathcal{R} = \{R, B, G\}$ that form nested spirals that pass multiple times through a thin region $\mathcal{I}$ of height 1 (formal definition below). The height of $\mathcal{I}$ is the bottleneck in the construction: it is traversed from left to right h times by each of R, B, and G. If we remove the parts of R, B, and G inside the region $\mathcal{I}$, we get $3h + 3$ connected components in total. Outside the the region $\mathcal{I}$, the three regions are more than $2h$ apart. This is illustrated in Figure 4 for $h = 3$. We formally define $\mathcal{I}$ to be the part of the plane within distance h of at least one of the bottom horizontal segments of the regions $\mathcal{R}$. All region components must be connected inside $\mathcal{I}$. Inside $\mathcal{I}$, it is possible that the grid polygons make different connections than those in $\mathcal{R}$. However, we argue that no matter how these connections are made, the grid polygons P_R, P_B, and P_G, together have to pass through $\mathcal{I}$ from left to right at least $h + 2$ times, thus requiring $\mathcal{I}$ to have height at least

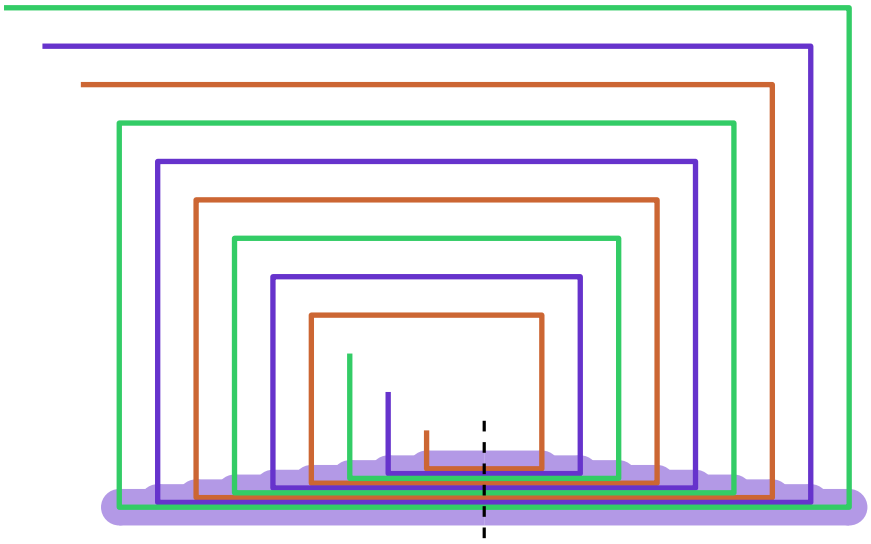


Figure 4 The regions for $h = 3$. The region $\mathcal{I}$ is highlighted. The dashed segment subdivides the boundary of $\mathcal{I}$ into its left and right part.

$2h + 3$. However, the available vertical space is only $2h + 1$ if the Hausdorff distance must stay below h, allowing $h + 1$ connections of pixel polygons. Hence, we obtain a contradiction.

The most involved part is to argue that P_R, P_B, and P_G, together have to pass through $\mathcal{I}$ at least $h + 2$ times. This argument critically depends on the following lemma.

▶ **Lemma 3.1.** *Given an alternating sequence $V = r_1, b_1, g_1, ..., r_k, b_k, g_k$ of $3k$ 3-colored points on a line ℓ, any planar drawing below the line ℓ connecting points of the same color induces a partition of the points into at least $2k + 1$ components.*

The line in Lemma 3.1 represents the left (or right) half of the boundary of $\mathcal{I}$. We can use the lemma to show that we can decrease the number of connected components by at most $h - 1$ by connecting the regions incident to the right side of $\mathcal{I}$ to other regions on the right side of $\mathcal{I}$. The same holds for the regions on the left side of $\mathcal{I}$. It thus follows that the remaining $3h + 3 - 2(h - 1) = h + 2$ components on the left side must be connected to the remaining $h + 2$ components on the right side of $\mathcal{I}$; after all, in the end there are only three regions left; one for each color. Therefore, P_R, P_B, and P_G pass through $\mathcal{I}$ at least $h + 2$ times as claimed. This allows us to obtain the following result:

▶ **Theorem 3.2.** *For any $h > 0$ there exist three regions $\mathcal{R} = \{R_1, R_2, R_3\}$, for which there is no valid assignment to grid polygons P_1, P_2, P_3 so that all regions $R_i \in \mathcal{R}$ have $H(R_i, P_i) < h$.*

4 Conclusion

In this paper we have shown what Hausdorff distance bounds can be attained when mapping disjoint simply-connected regions to the unit grid. We expressed our bounds in the number of regions. We have shown a worst case optimal bound of $O(m)$ for convex regions. For general regions there is a stark contrast between the bounds for two and three regions: for two regions it is constant, while for three regions it is unbounded. In the full paper we also show bounds on convex β-fat regions and point regions.

While we concentrated on worst-case optimal bounds, our constructive proof of the upper bound for convex regions will often give visually unfortunate output. Also, for a given

instance we will not achieve $O(1)$ Hausdorff distance even when it would be possible for that instance. This leads to the following two open problems. Firstly, can we realize visually reasonable output when this is possible for an instance (and how do we define this)? Secondly, can we realize a Hausdorff distance that is at most a constant factor worse than the best possible for each instance, in polynomial time?

References

1 Ernst Althaus, Friedrich Eisenbrand, Stefan Funke, and Kurt Mehlhorn. Point containment in the integer hull of a polyhedron. In *Proceedings 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 929–933, 2004.

2 Quirijn W Bouts, Irina Kostitsyna, Marc van Kreveld, Wouter Meulemans, Willem Sonke, and Kevin Verbeek. Mapping polygons to the grid with small Hausdorff and Fréchet distance. In *Proceedings 24th Annual European Symposium on Algorithms*, pages 22:1–22:16, 2016.

3 Man-Kwun Chiu and Matias Korman. High dimensional consistent digital segments. *SIAM Journal on Discrete Mathematics*, 32(4):2566–2590, 2018.

4 Man-Kwun Chiu, Matias Korman, Martin Suderland, and Takeshi Tokuyama. Distance bounds for high dimensional consistent digital rays and 2-d partially-consistent digital rays. *arXiv preprint arXiv:2006.14059*, 2020.

5 Tobias Christ, Dömötör Pálvölgyi, and Miloš Stojaković. Consistent digital line segments. *Discrete & Computational Geometry*, 47(4):691–710, 2012.

6 Jinhee Chun, Kenya Kikuchi, and Takeshi Tokuyama. Consistent digital curved rays. In *Abstracts 34th European Workshop on Computational Geometry*, 2019.

7 Jinhee Chun, Matias Korman, Martin Nöllenburg, and Takeshi Tokuyama. Consistent digital rays. *Discrete & Computational Geometry*, 42(3):359–378, 2009.

8 Mark de Berg, Dan Halperin, and Mark Overmars. An intersection-sensitive algorithm for snap rounding. *Computational Geometry*, 36(3):159–165, 2007.

9 Michael T. Goodrich, Leonidas J. Guibas, John Hershberger, and Paul J. Tanenbaum. Snap rounding line segments efficiently in two and three dimensions. In *Proceedings 13th Annual Symposium on Computational Geometry*, pages 284–293, 1997.

10 Leo J. Guibas and F. Frances Yao. On translating a set of rectangles. In *Proceedings 12th Annual ACM Symposium on Theory of Computing*, pages 154–160, 1980.

11 Warwick Harvey. Computing two-dimensional integer hulls. *SIAM Journal on Computing*, 28(6):2285–2299, 1999.

12 John Hershberger. Stable snap rounding. *Computational Geometry*, 46(4):403–416, 2013.

13 Reinhard Klette and Azriel Rosenfeld. *Digital Geometry: Geometric methods for digital picture analysis*. Elsevier, 2004.

14 Reinhard Klette and Azriel Rosenfeld. Digital straightness - a review. *Discrete Applied Mathematics*, 139(1-3):197–230, 2004.

15 Maarten Löffler and Wouter Meulemans. Discretized approaches to schematization. In *Proceedings 29th Canadian Conference on Computational Geometry*, 2017.

16 Arthur van Goethem, Irina Kostitsyna, Marc van Kreveld, Wouter Meulemans, Max Sondag, and Jules Wulms. The Painter's Problem: Covering a grid with colored connected polygons. In *Proceedings 25th International Symposium on Graph Drawing and Network Visualization*, pages 492–505, 2018.

A Dynamic Data Structure for k-Nearest Neighbors Queries

Sarita de Berg[1] **and Frank Staals**[1]

1 **Department of Information and Computing Sciences, Utrecht University, Netherlands**
s.deberg@uu.nl, f.staals@uu.nl

Abstract

We present an insertion-only data structure that supports k-nearest neighbors queries for a set of n point sites in $O(Q(n)\log n + k)$ time, based on any static data structure that can perform k'-nearest neighbors queries in $O(Q(n) + k')$ time. The key component is a general query algorithm that allows us to find k-nearest neighbors spread over t substructures simultaneously, thus reducing the $O(tk)$ term in the query time to $O(k)$. Applying this to the logarithmic method yields an insertion-only data structure with both efficient insertion and query time. We apply our method in the plane for the Euclidean and geodesic distance. We then briefly discuss the main difficulties to achieve a similar running time in the fully dynamic case.

1 Introduction

In the k-nearest neighbors (k-NN) problem we are given a set of n point sites S in $\mathbb{R}^d$, and we wish to preprocess these points such that for a query point q and an integer k, we can find the k sites in S 'closest' to q efficiently. This static problem has been studied in many different settings [3, 4, 8, 12, 13]. In particular, for sites in $\mathbb{R}^2$ and the Euclidean distance metric, Chan and Tsakalidis [8] achieved the optimal $O(\log n + k)$ query time using linear space and $O(n \log n)$ preprocessing time. Very recently, Liu showed how to achieve the same query time for general distance functions (in $\mathbb{R}^2$) using $O(n \log\log n)$ space [13].

In this paper, we study the dynamic version of the k-nearest neighbors problem, in which points can be inserted into or deleted from S, and the points lie in the plane. When we wish to report only one nearest neighbor (i.e. 1-NN searching), several efficient fully dynamic data structures exist [5, 7, 11]. Actually, all these data structures are variants of the same data structure by Chan [5]. For the Euclidean distance, the current best result using linear space achieves $O(\log^2 n)$ query time and polylogarithmic update time [7]. The variant by Kaplan et al. [11] achieves similar results for general distance functions. These data structures can also answer k-NN queries in $O(\log^2 n + k \log n)$ time [5]. Recently, Liu [13] claimed that the version of Kaplan et al. [11] can support such queries in $O(\log^2 n + k)$ time. However, we believe that there are some issues with this approach, as we briefly discuss in Section 4. For the Euclidean distance, Chan achieves a query time of $O(\log^2 n/\log\log n + k)$ for k-NN queries using $O(n \log n)$ space, by adapting his original data structure [6].

We are actually interested mostly in the insertion-only variant of the problem. Since nearest neighbor searching is decomposable, we can directly apply the logarithmic method [14] to turn a static k-NN searching data structure into an insertion-only data structure. However, this again yields an unwanted $O(k \log n)$ term in the query time. Our main goal is to reduce this term to $O(k)$ instead. In Section 2, we show how to achieve this goal. We present a general query algorithm that allows us to find the k-nearest neighbors spread over t substructures in $O(Q(n)t + k)$ time, assuming that the static data structure supports k'-NN

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

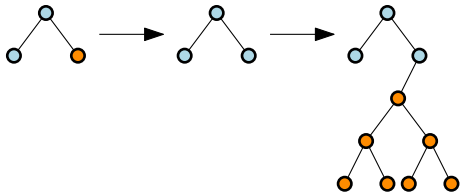


Figure 1 Example of expansion. Blue elements are included in a clan, orange elements are not. The expansion (building the next subheap) occurs when all elements have been included in a clan.

queries in $O(Q(n) + k')$ time. This yields a linear space data structure supporting queries in $O(\log^2 n + k)$ time and insertions in $O(\log^2 n)$ time, when using the Euclidean distance.[1]

Our original interest in the problem stems from a setting in which S is a set of points inside a simple polygon P with m vertices, and we use the geodesic distance as the distance measure. In this setting, Agarwal, Arge, and Staals [2] describe an insertion-only data structure for 1-NN queries that achieves $O(\log^2 n \log^2 m)$ query time, and $O(\log n \log^3 m)$ insertion time. As we show in Section 3, applying our machinery in this setting allows for efficient ($O(\log^2 n \log^2 m + k \log m)$ time) k-NN queries and insertions, as well.

2 Insertion-Only Data Structure

We describe a method that transforms a static k-NN data structure with query time $O(Q(n) + k)$ into an insertion-only k-NN data structure with query time $O(Q(n) \log n + k)$. Insertions take $O((P(n)/n) \log n)$ time, where $P(n)$ is the preprocessing time of the static data structure, and $C(n)$ is its space usage. We assume $Q(n)$, $P(n)$, and $C(n)$ are non-decreasing.

To support insertions, we use the logarithmic method [14]. We partition the sites into $O(\log n)$ groups $S_1, .., S_{O(\log n)}$ with $|S_i| = 2^i$ for $i \in \{1, .., O(\log n)\}$. To insert a site s, a new group containing only s is created. When there are two groups of size 2^i, these are removed and a new group of size 2^{i+1} is created. For each group we store the sites in the static k-NN data structure. This results in an amortized insertion time of $O((P(n)/n) \log n)$. This bound can also be made worst-case [14]. The main remaining issue is then how to support queries in $O(Q(n) \log n + k)$ time, thus avoiding an $O(k \log n)$ term in the query time.

Query algorithm. Let q be the query point and k the number of nearest neighbors we wish to find. We use the heap selection algorithm of Frederickson [9] to answer k-NN queries efficiently. This algorithm finds the k smallest elements of a binary min-heap of size $N \gg k$ in $O(k)$ time by forming groups of elements, called *clans*, in the original heap. Representatives of these clans are then added to another heap, and smaller clans are created from larger clans and organised in heaps recursively. For our purposes, we (only) need to consider how clans are formed in the original heap, because we do not build the entire heap we query before starting the algorithm. Instead, the heap is expanded during the query when necessary, see Figure 1 for an example. Note that any (non-root) element of the heap will only be included in a clan by the Frederickson algorithm after its parent has been included in a clan.

The heap H, on which we call the heap selection algorithm, contains all sites $s \in S$ exactly once, with the distance $d(s, q)$ as key for each site. Let $S_1, .., S_t$ be a partition of S

[1] With a slight variation of this method we can match the $O(\log^2 n / \log \log n + k)$ query time of Chan's [6] fully dynamic data structure for planes. However, this increases the insertion time to $O(\log^{2+\epsilon} / \log \log n)$.

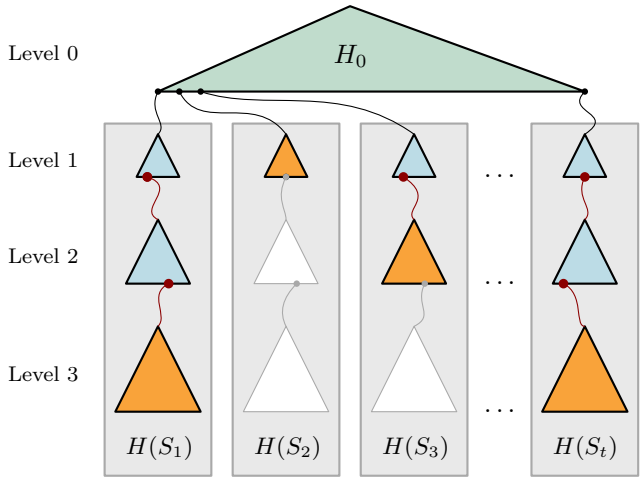


Figure 2 The heap that we construct for the k-nearest neighbors query. The subheaps of which all elements have been included in a clan are indicated in *blue*. The subheaps that have been built, but for which not all elements have been included in a clan, are indicated in *orange*. The *white* subheaps have not been built so far, because not all elements of their predecessor are in a clan yet.

into t disjoint sets. For each set of sites S_j, $j \in \{1, .., t\}$, we define a heap $H(S_j)$ containing all sites in S_j. We then "connect" these t heaps by building a dummy heap H_0 of size $O(t)$ that has the roots of all $H(S_j)$ as leaves. We set the keys of the elements of H_0 to $-\infty$. Let H be the complete data structure (heap) that we obtain this way, see Figure 2. It follows that we can now compute the k sites closest to q by finding the $|H_0| + k$ smallest elements in the resulting heap H and reporting only the non-dummy sites.

What remains is how to (incrementally) build the heaps $H(S_j)$ while running the heap selection algorithm. Each such heap consists of a hierarchy of *subheaps* $H_1(S_j), .., H_{O(\log n)}(S_j)$, such that every element of S_j appears in exactly one $H_i(S_j)$. Moreover, since the sets $S_1, .., S_j$ are pairwise disjoint, this holds for any $s \in S$, i.e. s appears in exactly one $H_i(S_j)$. Each heap $H_1(S_j)$ consists of the $k_1 = Q(n)$ sites in S_j closest to q, which we find by querying the static data structure of that group. We call these the *level 1* heaps. The subheap $H_i(S_j)$ at level $i > 1$ is built only after the last element e of $H_{i-1}(S_j)$ is included in a clan, i.e. e is considered by the heap selection algorithm. When e is included, we add a pointer from e to the root of $H_i(S_j)$, such that the root of $H_i(S_j)$ becomes a child of e, as in Figure 1.

To construct a subheap $H_i(S_j)$ at level $i > 1$, we query the static data structure of S_j using $k_i = k_1 2^{i-1}$. The new subheap is built using all sites returned by the query that have not been encountered earlier. It follows that all elements of $H_i(S_j)$ are larger than any of the elements in $H_1(S_j), .., H_{i-1}(S_j)$. Thus, the heap property is preserved.

Analysis of query time. As stated before, finding the k-smallest non-dummy elements of H takes $O(k + |H_0|)$ time [9]. In this section, we analyse the time used to construct H.

First, the level 0 and level 1 heaps are built. To build the level 1 heaps, we query each of the substructures using $k_1 = Q(n)$. In total these queries take $O((Q(n) + k_1)t) = O(Q(n)t)$ time. Building H_0 takes only $O(t)$ time. Retrieving the next k_i elements to build $H_i(S_j)$ for $i > 1$ requires a single query and thus takes $O(Q(n) + k_i)$ time. To bound the time used to build all heaps at level greater than 1, we first prove the following two lemmas.

▶ **Lemma 1.** *The size of a subheap $H_i(S_j)$, $j \in \{1, .., t\}$, at level $i > 1$ is exactly $k_1 2^{i-2}$.*

Proof. To create $H_i(S_j)$, we query the static data structure of S_j to find the $k_1 2^{i-1}$ sites closest to q. Of these sites, only the ones that have been not been included in any of the lower level subheaps are included in $H_i(S_j)$. The sites previously encountered are exactly the $k_1 2^{i-2}$ sites returned in the previous query. It follows that $|H_i(S_j)| = k_1(2^{i-1} - 2^{i-2}) = k_1 2^{i-2}$. ◀

▶ **Lemma 2.** *The total size of all subheaps $H_i(S_j)$ at level $i > 1$ is $O(k)$.*

Proof. There are essentially two types of subheaps: *complete* subheaps, of which all elements have been included in a clan (shown blue in Figure 2), and *incomplete* subheaps, of which only part of the elements has been included (shown orange in Figure 2). Note that the heap $H_i(S_j)$, $i > 1$, is only built when all elements of $H_{i-1}(S_j)$ have been included in a clan. In total, $O(k)$ elements (not in H_0) are included in a clan, so the total size of all complete subheaps is $O(k)$. Because the size of a subheap is at most twice the size of its predecessor, it follows that the total size of all incomplete heaps at level greater than 1 is also $O(k)$. ◀

Building $H_i(S_j)$ takes $O(Q(n)+k_i)$ time. To pay for this, we charge $O(1)$ to each element of $H_{i-1}(S_j)$. Because we choose $k_1 = Q(n)$, Lemma 1 implies that $|H_{i-1}(S_j)| = \Omega(Q(n))$, and that $k_i = k_1 2^{i-1} = 2^2 k_1 2^{i-3} = O(|H_{i-1}(S_j)|)$. From Lemma 2, and the fact that all subheaps are disjoint, it follows that we charge $O(1)$ to only $O(k)$ sites. We then have:

▶ **Lemma 3.** *Let $S_1, .., S_t$ be disjoint sets of point sites of sizes $n_1, .., n_t$, each stored in a data structure that supports k-NN queries in $O(Q(n_i) + k)$ time. There is a k-NN data structure on $\bigcup_i S_i$ that supports queries in $O(Q(n)t + k)$ time. The data structure uses $O(\sum_i C(n_i))$ space, where $C(n_i)$ is the space required by the k-NN structure on S_i.*

Applying Lemma 3 to the logarithmic method, we obtain the following result.

▶ **Theorem 4.** *Let S be a set of n point sites, and let $\mathcal{D}$ be a static k-NN data structure of size $O(C(n))$, that can be built in $O(P(n))$ time, and that can answer queries in $O(Q(n) + k)$ time. There is an insertion-only k-NN data structure on S of size $O(C(n))$ that supports queries in $O(Q(n) \log n + k)$ time. Inserting a new site in S takes $O((P(n)/n) \log n)$ time.*

Throughout this section, we used the standard assumption that for any two points p, q their distance $d(p, q)$ can be computed in constant time. When evaluating $d(p, q)$ takes T time, our technique achieves a query time of $O(T(Q(n) \log n + k))$.

3 Applications

Points in $\mathbb{R}^2$. In the Euclidean metric, k-nearest neighbors queries in the plane can be answered in $O(\log n + k)$ time, using $O(n)$ space and $O(n \log n)$ preprocessing time [1, 8].

▶ **Corollary 5.** *There is an insertion-only data structure of size $O(n)$ that stores a set of n sites in $\mathbb{R}^2$, allows for k-NNs queries in $O(\log^2 n + k)$ time, and insertions in $O(\log^2 n)$ time.*

By using the logarithmic method with only $O(\log_b n)$ groups, where $|S_i| = b^i$, we can improve the query time to $O(\log_b n \log n + k)$, at the cost of increasing the insertion time to $O(b \log_b n \log n)$. Setting $b = \log^\epsilon n$, we match the $O(\log^2 n / \log\log n + k)$ query time of Chan [6] and still achieve an insertion time of $O(\log^{2+\epsilon} / \log\log n)$. For general distance functions we achieve the same query time using Liu's data structure [13], using $O(n \log\log n)$ space and expected $O(\text{polylog}\, n)$ insertion time.

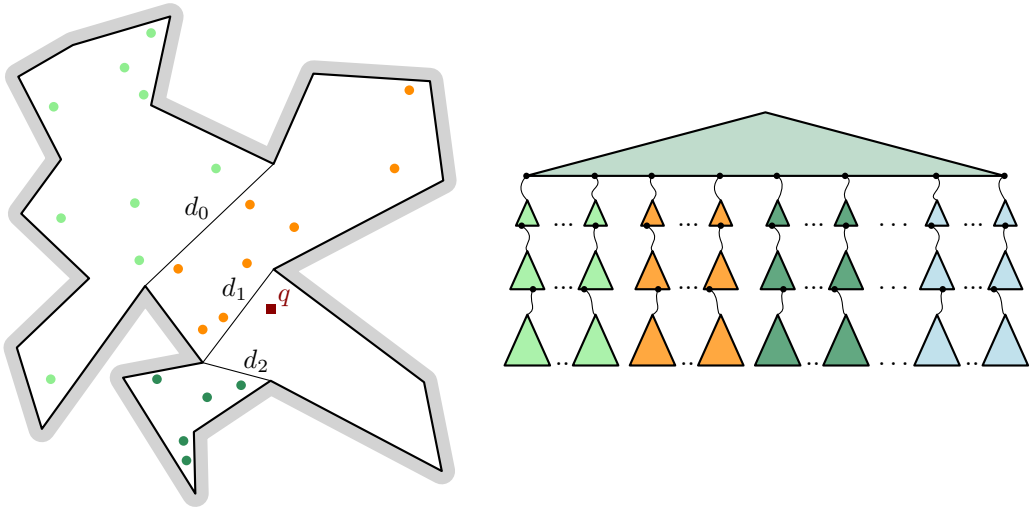


Figure 3 A partial decomposition of P and the corresponding heap used in a k-NN query for q.

Points in a simple polygon. In the geodesic k-nearest neighbors problem, S is a set of sites inside a simple polygon P with m vertices. For any two points p and q the distance $d(p, q)$ is defined as the length of the shortest path between p and q fully contained within P. The input polygon P can be preprocessed in $O(m)$ time so that the geodesic distance $d(p, q)$ between any two points $p, q \in P$ can be computed in $O(\log m)$ time [10].

We recursively partition the polygon P into two subpolygons P_r and P_ℓ of roughly the same size [2]. We denote by S_r and S_ℓ the sites in P_r and P_ℓ, respectively. This results in a decomposition of the polygon of $O(\log m)$ levels. For both sets, and at each of the levels, we again use the logarithmic method to support insertions. At every level, we store S_ℓ in the static k-NN query data structure of Theorem 22 of [2]. It requires $O(n \log n)$ space, excluding the size of the polygon, and finds the k-nearest neighbors among S_ℓ for a point $q \in P_r$ in $O((\log n + k) \log m)$ expected time. Building the data structure takes $O(n(\log n \log m + \log^2 m))$ time. Insertions using the logarithmic method therefore take $O(\log^2 n \log^2 m + \log n \log^3 m)$ time, as there are $O(\log m)$ levels in the decomposition of P.

During a k-NN query, we have a partition of S into $O(\log n \log m)$ disjoint groups S_j, as we consider one set of sites (S_ℓ or S_r) for each level of the decomposition. An example is shown in Figure 3. Lemma 3 thus states that there is a data structure that allows for k-NN queries in $O(\log^2 n \log^3 m + k \log m)$ time. We can reduce this by setting $k_1 = \log n$ instead of $\log n \log m$ and charging $O(\log m)$ to each site of $H_{i-1}(S_j)$ to pay for building $H_i(S_j)$.

▶ **Theorem 6.** *Let P be a simple polygon with m vertices. There is an insertion-only data structure of size $O(n \log n \log m + m)$ that stores a set of n point sites in P, allows for geodesic k-NN queries in $O(\log^2 n \log^2 m + k \log m)$ expected time, and inserting a site in $O(\log^2 n \log^2 m + \log n \log^3 m)$ time.*

4 Supporting Deletions

The data structures for 1-NN queries also supporting deletions in $O(\text{polylog}\, n)$ time are all based on an idea of Chan [5]. Liu [13] recently claimed that this data structure (in particular the version of Kaplan et al. [11]) also supports k-NN queries in $O(\log^2 n + k)$ time. We believe there are some issues with this approach, which we sketch below.

The key ingredient for dynamic 1-NN searching is an algorithm that takes a subset S of n of the sites and produces an (abstract) data structure $\mathcal{T}$ storing S, and a partition of S into a set "good" sites G and a set of "bad" sites B. The key properties are that every site in S is stored at most $O(\log n)$ times in $\mathcal{T}$, and that G has size $\Omega(n)$. By recursively applying this algorithm on the bad sites, we obtain a partition of S into $r = O(\log n)$ good sets $G_1, .., G_r$. Each good set G_i is stored in a static 1-NN searching data structure $\mathcal{D}_i$, and thus we can answer queries by querying each of these $O(\log n)$ data structures. Deleting a site may cause some sites in these good sets to become marked as bad. These sites are reinserted into the structure of B, hence we essentially move some sites from a G_i to a new good set G_j. It can be shown that the total number of good sets remains $O(\log n)$. When a query in some static data structure $\mathcal{D}_i$ returns a site marked as bad we simply discard it. Chan shows that this still allows us to answer queries correctly, and that deletions (and insertions) take $O(\text{polylog}\, n)$ amortized time [5, 11]. Note that the data structures $\mathcal{T}_1, .., \mathcal{T}_r$ are used only to collect which functions become bad when performing deletions, not to answer queries.[2]

Liu claims that this data structure can also support k-NN queries in $O(\log^2 n + k)$ time [13]. Presumably, by replacing the 1-NN data structures $\mathcal{D}_1, .., \mathcal{D}_r$ by k-NN data structures (all details are omitted). However, the sites in $\mathcal{D}_1, .., D_r$ are not pairwise disjoint, and thus we may encounter a site in the output to a query in multiple $\mathcal{D}_i$'s. This yields an $O(k \log n)$ term in the query time, which matches the bound given by Chan [5].

To answer k-NN queries efficiently, Chan adapted his original data structure to accomodate k-NN queries. By using the data structures $\mathcal{T}_1, .., \mathcal{T}_r$ to answer queries, and deleting planes that are removed from these structures explicitly, a query time of $O(\log^2 n / \log\log n + k)$ is achieved. However, it is not straightforward how to generalize this approach for more general distance functions (for example the geodesic distance function). We are currently working on this problem.

References

1 Peyman Afshani and Timothy M. Chan. Optimal halfspace range reporting in three dimensions. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 180–186. SIAM, 2009.

2 Pankaj K. Agarwal, Lars Arge, and Frank Staals. Improved dynamic geodesic nearest neighbor searching in a simple polygon. In *34th International Symposium on Computational Geometry, SoCG*, volume 99 of *LIPIcs*, pages 4:1–4:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

3 Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, 2008.

4 Timothy M. Chan. Random sampling, halfspace range reporting, and construction of $\leq k$-levels in three dimensions. *SIAM J. Comput.*, 30(2):561–575, 2000.

5 Timothy M. Chan. A dynamic data structure for 3-d convex hulls and 2-d nearest neighbor queries. *J. ACM*, 57(3):16:1–16:15, 2010.

6 Timothy M. Chan. Three problems about dynamic convex hulls. *Int. J. Comput. Geom. Appl.*, 22(4):341–364, 2012.

[2] In the presentation of Kaplan et al. [11] $\mathcal{D}_i$ actually coincides with the lowest "layer" of in the "tower" of shallow cuttings $\mathcal{T}_i$. The issue sketched here also applies to this version, as the cuttings in $\mathcal{T}_i$ do not cover the t-level of G_i but of $G_i \cup B$ for some B. The sites/functions in B are "good" w.r.t. some other G_j, and may thus be encountered multiple times.

7 Timothy M. Chan. Dynamic geometric data structures via shallow cuttings. In *35th International Symposium on Computational Geometry, SoCG*, volume 129 of *LIPIcs*, pages 24:1–24:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

8 Timothy M. Chan and Konstantinos Tsakalidis. Optimal deterministic algorithms for 2-d and 3-d shallow cuttings. *Discret. Comput. Geom.*, 56(4):866–881, 2016.

9 Greg N. Frederickson. An optimal algorithm for selection in a min-heap. *Inf. Comput.*, 104(2):197–214, 1993.

10 Leonidas J. Guibas and John Hershberger. Optimal shortest path queries in a simple polygon. *J. Comput. Syst. Sci.*, 39(2):126–152, 1989.

11 Haim Kaplan, Wolfgang Mulzer, Liam Roditty, Paul Seiferth, and Micha Sharir. Dynamic planar voronoi diagrams for general distance functions and their algorithmic applications. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 2495–2504. SIAM, 2017.

12 Der-Tsai Lee. On k-nearest neighbor voronoi diagrams in the plane. *IEEE Transactions on Computers*, C-31(6):478–487, June 1982.

13 Chih-Hung Liu. Nearly optimal planar k nearest neighbors queries under general distance functions. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 2842–2859. SIAM, 2020.

14 Mark H. Overmars. *The Design of Dynamic Data Structures*, volume 156 of *Lecture Notes in Computer Science*. Springer, 1983.

Tukey Depth Histograms*

Daniel Bertschinger[1], Jonas Passweg[2], and Patrick Schnider[3]

1 **Department of Computer Science, ETH Zürich, Switzerland**
`daniel.bertschinger@inf.ethz.ch`
2 **Department of Computer Science, ETH Zürich, Switzerland**
`jpassweg@student.ethz.ch`
3 **Department of Mathematical Sciences, University of Copenhagen**
`ps@math.ku.dk`

Abstract

The Tukey depth of a flat with respect to a point set is a concept that appears in many areas of discrete and computational geometry. In this work, we introduce the *Tukey depth histogram* of k-flats in $\mathbb{R}^d$ with respect to a point set P, which is a vector $D^{k,d}(P)$, whose i'th entry $D^{k,d}_i(P)$ denotes the number of k-flats spanned by $k+1$ points of P that have Tukey depth i with respect to P. We give a complete characterization of the depth histograms of points, that is, for any dimension d we give a description of all possible histograms $D^{0,d}(P)$.

Related Version A full version is available at `http://arxiv.org/abs/2103.08665`

1 Introduction

Many fundamental problems on point sets, such as the number of extreme points, the number of halving lines, or the crossing number do not depend on the actual location and distances of the points, but rather on some underlying combinatorial structure of the point set. There is a vast body of work of combinatorial representations of point sets, at the beginning of which are the seminal series of papers by Goodman and Pollack [2, 3, 4], where many important objects such as *allowable sequences* and *order types* are introduced. In particular order types have proven to be a very powerful representation of point sets. For many problems however, less information than what is encoded in order types is sufficient. One example for such a problem is the determination of the *Tukey depth* of a query point with respect to a planar point set. The Tukey depth of a query point q with respect to a point set P is the minimum number of points of P that lie in a closed halfspace containing q. In the plane, this can be computed knowing only for each k, how many directed lines through q and a point of P have exactly k points to their left. This defines the *ℓ-vector* of q. The Tukey depth of q is now just the smallest k for which the corresponding entry in the ℓ-vector is non-zero. In [6], a characterization of all possible ℓ-vectors is given, phrased in terms of *frequency vectors*, which is an equivalent object.

Interesting objects emerge after forgetting yet another piece of information: instead of knowing the ℓ-vector of each point, assume we only know the sum of all ℓ-vectors. This corresponds to knowing for each j the number of *j-edges* that is, knowing the histogram of j-edges. The number of j-edges that a point set admits is a fundamental question in discrete geometry and has a rich history, see e.g. [7], Chapter 4 in [1] or Chapter 11 in [5] and the references therein.

In this work, we investigate a similar concept: depth histograms of points. This corresponds to knowing for each j how many points of Tukey depth j are in the point set. We

* The third author has received funding from the European Research Council under the European Unions Seventh Framework Programme ERC Grant agreement ERC StG 716424 - CASe.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

give a complete characterization of possible such histograms for point sets in general position. In particular, we will show the following:

▶ **Theorem 1.1.** *A vector $D^{0,d}$ is a depth histogram of a point set in general position in $\mathbb{R}^d$ if and only if for all nonzero entries $D_i^{0,d}$ with $i \geq 2$ we have*

$$\sum_{j=1}^{i-1} D_j^{0,d} \geq 2i + d - 3.$$

In fact, both depth histograms of points as well as j-edges can be viewed as instances of a more general definition, that of histograms of j-flats, which we will introduce in the following. We hope that this work can serve as a small step in the systematic study of these histograms.

In order to define histograms of j-flats, we first define the Tukey depth of a flat:

▶ **Definition 1.2.** Let Q be a set of $k+1$ points in $\mathbb{R}^d$, $k < d$, which span a unique k-flat F. The *affine Tukey depth* of Q with respect to a point set P, denoted by $atd_P(Q)$, is the minimum number of points of P in any closed halfspace containing F. The *convex Tukey depth* of Q with respect to P, denoted by $ctd_P(Q)$, is the minimum number of points of P in any closed halfspace containing $\mathrm{conv}(Q)$.

Note that for $k = 0$ both definitions coincide with the standard definition of Tukey depth, and we just write $td_P(q)$ in this case. Further note that if $P \cup Q$ is in convex position, then $atd_P(Q) = ctd_P(Q)$.

▶ **Definition 1.3.** Let P be a set of points in $\mathbb{R}^d$. The *affine Tukey depth histogram of j-flats*, denoted by $D^{j,d}(P)$, is a vector whose entries $D_l^{j,d}(P)$ are the number of subsets $Q \subset P$ of size $j+1$ whose affine Tukey depth is i. Similarly, replacing affine Tukey depth with convex Tukey depth, we define the *convex Tukey depth histogram of j-flats*, denoted by $cD^{j,d}(P)$.

In the following, we will also call affine Tukey depth histograms just *depth histograms*, that is, unless we specify the *convex*, we always mean an affine Tukey depth histogram. Note however that for $j = 0$ or if P is in convex position, the two histograms coincide.

Many problems in discrete geometry can be phrased in terms of depth histograms. For example, the number of extreme points of a point set P just corresponds to the entry $D_1^{0,d}(P)$ (note that each point of P has Tukey depth at least 1). Further, the number of j-edges or, more generally, j-facets corresponds to the entry $D_j^{d-1,d}(P)$.

2 The condition is necessary

▶ **Lemma 2.1.** *For any point set $P \subseteq \mathbb{R}^d$ and any point $p \in P$ we have $\mathrm{td}_P(p) \leq \frac{n-d+2}{2}$.*

Sketch of proof. Let $P \subseteq \mathbb{R}^d$ and let $p \in P$ be any point with $td_P(p) = k$. We show that any such point set consists of at least $2k - 2 + d$ points, which proves the lemma. Consider a witnessing halfspace h_p of p and its bounding hyperplane h. After rotation and translation, we may assume that h contains p and d other points of P, and h_p contains $k-1$ points in its interior. The same has to hold for the complement of h_p, giving at least $2(k-1)+d$ points. ◀

▶ **Lemma 2.2.** *For any point set $P \subseteq \mathbb{R}^d$ and any two points $p, q \in P$ with $\mathrm{td}_P(p) \leq \mathrm{td}_P(q)$ we have $\mathrm{td}_P(p) = \mathrm{td}_{P \setminus q}(p)$.*

Proof. The point q cannot lie in any witnessing halfspace for p. ◀

By repeatedly applying the lemma one can easily show the following.

▶ Proposition 1. For any depth histogram $[a_1, a_2, \ldots, a_{m-1}, a_m]$, both $[a_1, a_2, \ldots, a_{i-1}, a_i]$ and $[a_1, a_2, \ldots, a_{i-1}, 1]$ with $i \leq m$ are depth histograms.

▶ **Corollary 2.3** (Necessary condition of Theorem 1.1). *For any depth histogram $D^{0,d}$ and all nonzero entries $D^{0,d}_i$ with $i \geq 2$ we have*

$$\sum_{j=1}^{i-1} D^{0,d}_j \geq 2i + d - 3.$$

Proof. For the sake of contradiction, let us assume that there is a depth histogram $D^{0,d}$ for which there is a nonzero entry $D^{0,d}_i$ and $\sum_{j=1}^{i-1} D^{0,d}_j < 2i + d - 3$. Using Proposition 1, we can cut off the depth histogram $D^{0,d}$ at any point and so we can consider the histogram $D' := [D^{0,d}_1, \ldots, D^{0,d}_{i-1}, 1]$. But then for the point set P' corresponding to this histogram, we have $|P'| < 2i + d - 2$, which contradicts Lemma 2.1. ◀

2.1 Two special configurations

We make a small detour and revisit Lemma 2.1, noting that the given bound is tight. We will show this using point sets in so-called *symmetric configuration* [6]. These point sets will also be at the core of our proof that the condition of Theorem 1.1 is sufficient.

▶ **Definition 2.4.** A point set $P \subseteq \mathbb{R}^d$ in general position is in

1. *symmetric configuration* if and only if there exists a *central point* $c \in P$ such that every hyperplane through c and $d-1$ other points of P separates the remaining points into two halves of equal size.
2. *eccentric configuration* if and only if there exists a *central point* $c \in P$ such that every hyperplane through c and $d-1$ other points of P *almost* separates the remaining points into two halves of equal size, that is, divides the remaining points in two sets with difference in cardinality of at most 1.

Note that depending on the dimension and the size of P, only one of the definitions can be applied. Examples of such point sets are given in Figure 1.

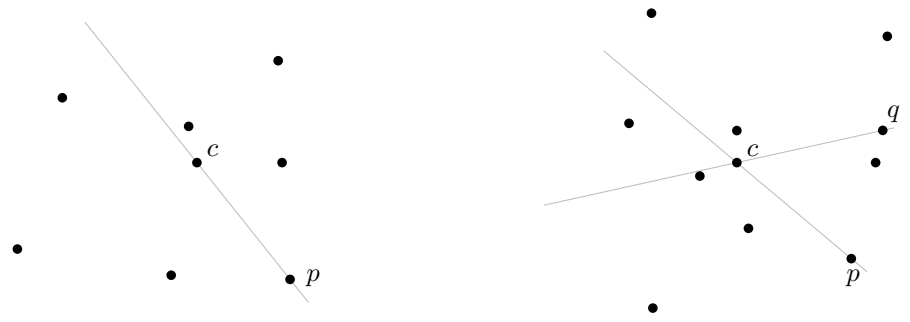


Figure 1 Two point sets in symmetric and eccentric configuration, respectively. The lines through c and p or c and q, respectively, (almost) divide the remaining point set.

The following follows from the definition:

▶ **Lemma 2.5.** *The symmetric central point c in a symmetric (or eccentric) point set P has depth* $\mathrm{td}_P(c) = \lfloor \frac{n-d+2}{2} \rfloor$.

At first glance, it is not clear that symmetric and eccentric point sets of any size exist in any dimension. We will show that they do in the next section, this will be an important step in proving that the condition of Theorem 1.1 is sufficient. For space reasons, we only sketch the argument for higher dimensions.

3 The condition is sufficient

To prove that the condition we gave in Theorem 1.1 is sufficient, we build up point sets according to their histograms by adding points one-by-one. In other words, given a histogram, we start with points in convex position (as many as there are of depth 1). We then add new points in the "center" of the point set, push them outwards until they have the right depth, without changing the depth of any other point, maintaining that we can always add yet another point in the center to get a symmetric or eccentric point set.

3.1 Moving points

Note that the Tukey depth of q can only change if q is involved in a change in the order type. In other words, q was pushed over a hyperplane formed by d other points of the point set. We now formally characterize what happens in any such case.

▶ Proposition 2. Let $P \in \mathbb{R}^d$ be a point set and $q \in P$ be an arbitrary point. Let q' be a point close to q, such that the order types of P and $P' := P \setminus \{q\} \cup \{q'\}$ only differ in one simplex $\mathcal{S}$, that is, $\mathcal{S} := conv\{p_1, \ldots, p_d, q\}$ and $\mathcal{S}'$, respectively. Let h be the hyperplane spanned by $p_1, \ldots, p_d$ and $\hat{q}$ the intersection of h with the line qq'.

- If $\hat{q} \notin conv\{p_1, \ldots, p_d\}$, then $td_P(q) = td_{P'}(q')$, and
- otherwise, if $\hat{q} \in conv\{p_1, \ldots, p_d\}$, then $|td_P(q) - td_{P'}(q')| \leq 1$.

For a proof, we refer to the full version of this paper. Note that whenever q has the highest depth among all points, we also know that the depths of the other points do not change.

▶ Observation 3.1. Whenever we have $td_P(q) > td_P(p)$ for all points p in the point set, then $td_P(p) = td_{P'}(p)$.

3.2 Inserting a new point

We have already seen point sets, that contain a point of maximum possible depth. These special point sets will help us placing new points of large depth, which we then can push outwards. For this, let P be a point set in general position and in symmetric (eccentric, respectively) configuration missing the symmetric central point. If we place a new point p at the location of the (previously inexistent) symmetric central point, then by Lemmas 2.1 and 2.5, we know that p has the maximal possible depth. Now, we are able to push p outwards until it has the desired depth and the resulting point set is in eccentric (symmetric, respectively) configuration. An example in dimension 2 can be found in Figure 2.

▶ **Lemma 3.2.** *For any point set $P \subseteq \mathbb{R}^2$ in general position and in eccentric (symmetric, resp.) configuration there exists a direction in which we can push the central point such that after adding a new center we have a symmetric (eccentric, resp.) point set in general position.*

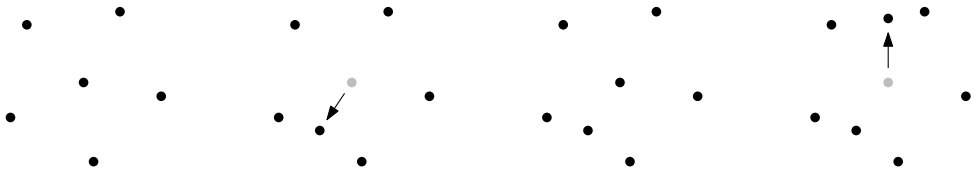


Figure 2 A point set in symmetric configuration (left). After pushing the symmetric central point out (second from left), we arrive at a point set in eccentric configuration missing the symmetric central point. Adding a new point at maximum possible depth (third from left). Pushing out again gets us back into a symmetric point set missing the symmetric central point (rightmost).

Proof. First, note that if P is in symmetric configuration, any direction does the job. If P is eccentric, then there exist two neighbors p_1 and p_2 in the rotational order of points around q without a symmetric central line dividing them. Now choose to move q outwards on an "opposite" halfline, see Figure 3, right. ◀

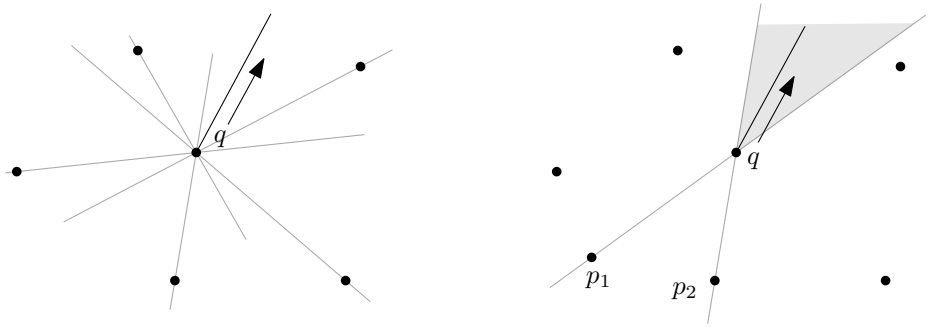


Figure 3 The central point and the direction in which we push it if the point set is symmetric (left) and if the point set is eccentric (right).

In higher dimensions it is not so easy to see how to get the directions and why they always exist. The core of our proof is the following theorem, whose full proof can be found in the full version of the paper:

▶ **Theorem 3.3.** *For every symmetric point set $P \subseteq \mathbb{R}^d$, there exist two directions v_1 and v_2 such that we can push the central point into either direction; add a new central point and arrive at an eccentric point set P'. We can then push the newly added point into the other direction, and arrive at a symmetric point set P'' missing the symmetric central point.*

Sketch of proof. Project P to the sphere $S^{d-1} \subseteq \mathbb{R}^d$ and note that the resulting point P' set is still symmetric. Assume without loss of generality that no point of P' is at the north or the south pole. If we choose v_1 and v_2 as the directions to the north and the south pole, respectively, we are almost done; the only issue is that the resulting point set is not in general position. However, using stereographic projection at the north pole, we get an eccentric point set $Q \subseteq \mathbb{R}^{d-1}$ whose central point is the projection of the north pole. Using induction on the dimension, we may assume that we can move the projection of the north pole into some direction and add a new central point c so that the resulting point set Q' is symmetric. Reversing the stereographic projection, and placing the new central point c at the south pole, the directions v_1 and v_2 are given by the moved north pole and the south pole. ◀

4 Conclusion

We have introduced Tukey depth histograms of j-flats, which relate to several problems in discrete geometry. For histograms of points, we were able to give a full characterization.

It is an interesting open problem to find better necessary and also sufficient conditions, perhaps even characterizations, of histograms of j-flats for $j > 0$. We hope that the ideas and arguments in this paper might be useful in this endeavor.

Another interesting open problem is to relate depth histograms to other representations of point sets. For example in the plane, the order type determines the ℓ-vectors for each point, but not vice-versa, that is, there are point sets that have the same sets of ℓ-vectors but different order types. Similarly, the set of ℓ-vectors determines the histograms $D^{0,2}$ and $D^{1,2}$. Is it true that the reverse is also true or are there point sets for which both $D^{0,2}$ and $D^{1,2}$ are the same but whose sets of ℓ-vectors are different?

References

1 Stefan Felsner. *Geometric graphs and arrangements: some chapters from combinatorial geometry*. Springer Science & Business Media, 2012.

2 Jacob Goodman and Richard Pollack. A theorem of ordered duality. *Geometriae Dedicata*, 12:63–74, 01 1982. doi:10.1007/BF00147331.

3 Jacob Goodman and Richard Pollack. Multidimensional sorting. *SIAM J. Comput.*, 12:484–507, 08 1983. doi:10.1137/0212032.

4 Jacob Goodman and Richard Pollack. Semispaces of configurations, cell complexes of arrangements. *J. Comb. Theory, Ser. A*, 37:257–293, 11 1984. doi:10.1016/0097-3165(84)90050-5.

5 Jiri Matousek. *Lectures on discrete geometry*, volume 212. Springer Science & Business Media, 2013.

6 A. J. Ruiz-Vargas and E. Welzl. Crossing-free perfect matchings in wheel point sets. In *A Journey Through Discrete Mathematics: A Tribute to Jiří Matoušek*, pages 735–764. 2017. doi:10.1007/978-3-319-44479-6_30.

7 Uli Wagner. k-sets and k-facets. *Contemporary Mathematics*, 453:443, 2008.

Enclosing Depth and other Depth Measures*

Patrick Schnider[1]

1 **Department of Mathematical Sciences, University of Copenhagen**
ps@math.ku.dk

Abstract

We study families of depth measures defined by natural sets of axioms. We show that any such depth measure is a constant factor approximation of Tukey depth. Along the way, we introduce and study a new depth measure called *enclosing depth*, which we believe to be of independent interest, and show its relation to a constant-fraction Radon theorem on certain two-colored point sets.

Related Version A full version is available at http://arxiv.org/abs/2103.08421

1 Introduction

Medians are an important tool in the statistical analysis and visualization of data. Various generalizations of medians to higher dimensions have been introduced and studied, see e.g. [1, 9, 12] for surveys. Many of these generalized medians rely on a notion of *depth* of a query point within a data set, a median then being a query point with the highest depth among all possible query points. In particular, just like the median, many of these depth measures only depend on the relative positions of the involved points, making them robust against outliers. More formally, let $S^{\mathbb{R}^d}$ denote the family of all finite sets of points in $\mathbb{R}^d$. A depth measure is a function $\varrho : (S^{\mathbb{R}^d}, \mathbb{R}^d) \to \mathbb{R}_{\geq 0}$ which assigns to each pair (S, q) consisting of a finite set of data points S and a query point q a value, which describes how deep the query point q lies within the data set S. We call a depth measure ϱ *combinatorial* if it depends only on the order type of $S \cup \{q\}$. In this paper, we consider general classes of combinatorial depth measures, defined by a small set of axioms, and prove relations between them and concrete depth measures, such as *Tukey depth* (TD) and *Tverberg depth* (TvD).

▶ **Definition 1.1.** Let S be a finite point set in $\mathbb{R}^d$ and let q be a query point. Then the Tukey depth of q with respect to S, denoted by $\mathbf{TD}(S, q)$, is the minimum number of points of S in any closed halfspace containing q.

Tukey depth was introduced by John W. Tukey in 1975 [15] and has received significant attention since, both from a combinatrial as well as from an algorithmic perspective, see e.g. Chapter 58 in [14] and the references therein. Notably, the *centerpoint theorem* states that for any point set $S \subset \mathbb{R}^d$, there exists a point $q \in \mathbb{R}^d$ for which $\mathrm{TD}(S, q) \geq \frac{|S|}{d+1}$ [13].

In order to define Tverberg depth, we need a preliminary definition: given a point set S in $\mathbb{R}^d$, an *r-partition* of S is a partition of S into r pairwise disjoint subsets $S_1, \dots, S_r \subset S$ with $\bigcap_{i=1}^{r} \mathrm{conv}(S_i) \neq \emptyset$. We call $\bigcap_{i=1}^{r} \mathrm{conv}(S_i)$ the *intersection* of the r-partition.

▶ **Definition 1.2.** Let S be a finite point set in $\mathbb{R}^d$ and let q be a query point. Then the Tverberg depth of q with respect to S, denoted by $\mathbf{TvD}(S, q)$, is the maximum r such that there is an r-partition of S whose intersection contains q.

* The author has received funding from the European Research Council under the European Unions Seventh Framework Programme ERC Grant agreement ERC StG 716424 - CASe. Part of this work was done when the author was employed at ETH Zürich.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Tverberg depth is named after Helge Tverberg who proved in 1966 that any set of $(d+1)(r-1)+1$ points in $\mathbb{R}^d$ allows an r-partition [16]. In particular, this implies that there is a point q with $\mathrm{TvD}(S,q) \geq \frac{|S|}{d+1}$. Just as for Tukey depth, there is an extensive body of work on Tverbergs theorem, see the survey [3] and the references therein.

In $\mathbb{R}^1$, both Tukey and Tverberg depth give a very natural depth measure: it counts the number of points of S to the left and to the right of q and then returns the minimum of the two numbers. We call this measure the *standard depth* in $\mathbb{R}^1$.

Another depth measure that is important in this paper is called *enclosing depth*. We say that a point set S of size $(d+1)k$ in $\mathbb{R}^d$ *k-encloses* a point q if S can be partitioned into $d+1$ pairwise disjoint subsets $S_1, \ldots, S_{d+1}$, each of size k, in such a way that for every transversal $p_1 \in S_1, \ldots, p_{d+1} \in S_{d+1}$, the point q is in the convex hull of $p_1, \ldots, p_{d+1}$. Intuitively, the points of S are centered around the vertices of a simplex with q in its interior.

▶ **Definition 1.3.** Let S be a finite point set in $\mathbb{R}^d$ and let q be a query point. Then the enclosing depth of q with respect to S, denoted by $\mathbf{ED}(S,q)$, is the maximum k such that there exists a subset of S which k-encloses q.

It is straightforward to see that enclosing depth also gives the standard depth in $\mathbb{R}^1$. The centerpoint theorem [13] and Tverberg's theorem [16] show that both for Tukey as well as Tverberg depth, there are deep points in any dimension. We will show that this also holds for enclosing depth. In fact, we will show that enclosing depth can be bounded from below by a constant fraction of Tukey depth. From this we get the main results of this paper: all depth measures that satisfy the axioms given later are a constant factor approximation of Tukey depth.

2 A first set of axioms

The first set of depth measures that we consider are *super-additive* depth measures. A combinatorial depth measure $\varrho : (S^{\mathbb{R}^d}, \mathbb{R}^d) \to \mathbb{R}_{\geq 0}$ is called super-additive if it satisfies the following conditions:

(i) for all $S \in S^{\mathbb{R}^d}$ and $q, p \in \mathbb{R}^d$ we have $|\varrho(S,q) - \varrho(S \cup \{p\}, q)| \leq 1$ (sensitivity),
(ii) for all $S \in S^{\mathbb{R}^d}$ and $q \in \mathbb{R}^d$ we have $\varrho(S,q) = 0$ for $q \notin \mathrm{conv}(S)$ (locality),
(iii) for all $S \in S^{\mathbb{R}^d}$ and $q \in \mathbb{R}^d$ we have $\varrho(S,q) \geq 1$ for $q \in \mathrm{conv}(S)$ (non-triviality),
(iv) for any disjoint subsets $S_1, S_2 \subseteq S$ and $q \in \mathbb{R}^d$ we have $\varrho(S,q) \geq \varrho(S_1,q) + \varrho(S_2,q)$ (super-additivity).

It is not hard to show that a one-dimensional depth measure which satisfies these conditions has to be the standard depth measure (in fact, the arguments are generalized to higher dimensions in the following two observations) and that no three conditions suffice for this. Further, it can be shown that both Tukey depth and Tverberg depth are super-additive. The following two observations follow from the definitions, see the full version for the proofs:

▶ Observation 2.1. For every depth measure ϱ satisfying (i) sensitivity and (ii) locality and for all $S \in S^{\mathbb{R}^d}$ and $q \in \mathbb{R}^d$ we have $\varrho(S,q) \leq \mathrm{TD}(S,q)$.

▶ Observation 2.2. For every depth measure ϱ satisfying (iii) non-triviality and (iv) super-additivity and for all $S \in S^{\mathbb{R}^d}$ and $q \in \mathbb{R}^d$ we have $\varrho(S,q) \geq \mathrm{TvD}(S,q)$.

Finally, it is not too hard to show that $\mathrm{TvD}(S,q) \geq \frac{1}{d}\mathrm{TD}(S,q)$, see e.g. [7] for an argument. Combining these observations, we thus get the following.

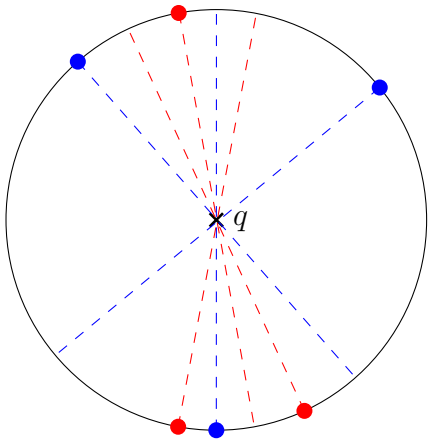


Figure 1 Enclosing depth does not satisfy the super-additivity condition: the point q has enclosing depth 1 with respect to both the blue and the red points, but its enclosing depth with respect to the union of the two sets is still 1.

▶ **Corollary 2.3.** *Let ϱ be a super-additive depth measure. Then for every point set S and query point q in $\mathbb{R}^d$ we have*

$$TD(S,q) \geq \varrho(S,q) \geq TvD(S,q) \geq \frac{1}{d} TD(S,q).$$

From Corollary 2.3 it follows that for any super-additive depth measure and any point set there is always a point of depth at least $\frac{|S|}{d+1}$, for example any Tverberg point. On the other hand, there are depth measures that give the standard depth in $\mathbb{R}^1$ that are not super-additive, for example enclosing depth, see Figure 1.

3 A second set of axioms

The second family of depth measures we consider are *central* depth measures. A combinatorial depth measure $\varrho : (S^{\mathbb{R}^d}, \mathbb{R}^d) \to \mathbb{R}_{\geq 0}$ is called central if it satisfies the following conditions:

(i) for all $S \in S^{\mathbb{R}^d}$ and $q, p \in \mathbb{R}^d$ we have $|\varrho(S,q) - \varrho(S \cup \{p\}, q)| \leq 1$ (sensitivity),
(ii) for all $S \in S^{\mathbb{R}^d}$ and $q \in \mathbb{R}^d$ we have $\varrho(S,q) = 0$ for $q \notin \text{conv}(S)$ (locality),
(iii') for every $S \in S^{\mathbb{R}^d}$ there is a $q \in \mathbb{R}^d$ for which $\varrho(S,q) \geq \frac{1}{d+1}|S|$ (centrality).
(iv') for all $S \in S^{\mathbb{R}^d}$ and $q, p \in \mathbb{R}^d$ we have $\varrho(S \cup \{p\}, q) \geq \varrho(S,q)$ (monotonicity),

We have seen before that any super-additive depth measure indeed satisfies the centrality condition, and the super-additivity condition (iv) is stronger than the monotonicity condition (iv'), so central depth measures are a superset of super-additive depth measures. It is actually a strict superset, as for example the depth measure whose depth regions are defined as the convex hulls of Tverberg depth regions is central but not super-additive.

While central depth measures enforce deep points by definition, they might still differ from each other a lot locally. In the following, we will bound by how much they differ locally, showing that every central depth measure is a constant factor approximation of Tukey depth.

▶ **Theorem 3.1.** *Let ϱ be a central depth measure in $\mathbb{R}^d$. Then there exists a constant $c_1 = c_1(d)$, which depends only on the dimension d, such that*

$$TD(S,q) \geq \varrho(S,q) \geq ED(S,q) - (d+1) \geq c_1 \cdot TD(S,q) - (d+1).$$

Here the first inequality is just Observation 2.1. As for the second inequality, we only give a sketch here and refer the interested reader to the full version. We would like to argue that if S k-encloses q then $\varrho(S,q) = k$. By centrality, there must indeed be a point q' with $\varrho(S,q') = k$ (note that $|S| = k(d+1)$ by definition of k-enclosing), but this point can lie anywhere in the centerpoint region of S and not every point in the centerpoint region is k-enclosed by S. However, by adding $d+1$ points very close to q, we can ensure that q is the only possible centerpoint in the new point set, and the second inequality then follows from sensitivity and monotonicity after removing these points again. The most involved part of Theorem 3.1 is the last inequality, which we will now prove:

▶ **Theorem 3.2** ($E(d)$)**.** *There is a constant $c_1 = c_1(d)$ such that for all $S \in S^{\mathbb{R}^d}$ and $q \in \mathbb{R}^d$ we have $ED(S,q) \geq c_1 \cdot TD(S,q)$.*

Note that $E(1)$ is true and $c_1(1) = 1$. Let $P = R \cup B$ be a bichromatic point set with color classes R (red) and B (blue). We say that B *surrounds* R if for every halfspace h we have $|B \cap h| \geq |R \cap h|$. Note that this in particular implies $|B| \geq |R|$. The statement $E(d)$ is related to the following constant-fraction Radon theorem:

▶ **Theorem 3.3** ($R(d)$)**.** *Let $P = R \cup B$ be a bichromatic point set in $\mathbb{R}^d$ where B surrounds R. Then there is a constant $c_2 = c_2(d)$ such that there are integers a and b and pairwise disjoint subsets $R_1, \ldots, R_a \subseteq R$ and $B_1, \ldots, B_b \subseteq B$ with*

1. *$a + b = d + 2$,*
2. *$|R_i| \geq c_2 \cdot |R|$ for all $1 \leq i \leq a$,*
3. *$|B_i| \geq c_2 \cdot |R|$ for all $1 \leq i \leq b$,*
4. *for every transversal $r_1 \in R_1, \ldots, r_a \in R_a, b_1 \in B_1, \ldots, b_b \in B_b$, we have $conv(r_1, \ldots, r_a) \cap conv(b_1, \ldots, b_b) \neq \emptyset$.*

In other words, the Radon partition respects the color classes. It can be shown that $R(1)$ can be satisfied choosing $a = 1$, $b = 2$ and $c_2(1) = \frac{1}{3}$, see the appendix for a proof. In the following, we will prove that $R(d-1) \Rightarrow E(d)$ and that $E(d-1) \Rightarrow R(d)$. By induction, these two claims then imply Theorem 3.1.

▶ **Lemma 3.4.** *$R(d-1) \Rightarrow E(d)$.*

Sketch of Proof. Assume without loss of generality that q is the origin an that the halfspace $h : x_d \leq 0$ witnesses $\mathrm{TD}(S,q) = k$. Consider the point set S' derived from S by central projection through q to the hyperplane $x_d = 1$, and color all points from h red and the points from the complement h^c blue. Then S' is a $(d-1)$-dimensional point set where B surrounds R. Further, every Radon partition in S' which respects the color classes corresponds to a simplex in S which contains q. ◀

For the proof of the second implication, we need to recall a few results, starting with the *Same Type Lemma* by Bárány and Valtr [4].

▶ **Theorem 3.5** (Theorem 2 in [4])**.** *For every two natural numbers d and m there is a constant $c_3(d,m) > 0$ with the following property: Given point sets $X_1, \ldots, X_m \subseteq \mathbb{R}^d$ such that $X_1 \cup \ldots \cup X_m$ is in general position, there are subsets $Y_i \subseteq X_i$ with $|Y_i| \geq c_3 \cdot |X_i|$ such that all transversals of the Y_i have the same order type.*

The second result that we will need is the *Center Transversal Theorem*, proved independently by Dol'nikov [6] as well as Zivaljević and Vrećica [17]. We will only need the version for two colors, so we state it in this restricted version:

▶ **Theorem 3.6** (Center Transversal for two colors). *Let μ_1 and μ_2 be two finite Borel measures on $\mathbb{R}^d$. Then there exists a line ℓ such that for every closed halfspace H which contains ℓ and every $i \in \{1, 2\}$ we have $\mu_i(H) \geq \frac{\mu_i(\mathbb{R}^d)}{d}$.*

Such a line ℓ is called a *center transversal*. By a standard argument (replacing points with balls of small radius, see e.g. [10]), the same result also holds for two point sets P_1, P_2 in general position, where $\mu_i(H)$ is replaced by $|P_i \cap H|$. The lemma that is at the core of the proof of the center transversal theorem is the following, again proved independently by Dol'nikov [6] as well as Zivaljević and Vrećica [17]:

▶ **Lemma 3.7.** *Let g_1 and g_2 be two continuous assignments of points to the set of all $(d-1)$-dimensional linear subspaces of $\mathbb{R}^d$. Then there exists such a subspace F in which $g_1(F) = g_2(F)$.*

The centerpoint theorem follows by choosing in a unique way g_1 and g_2 in the centerpoint region of projected masses. If the two measures can be separated by a hyperplane, we can do something similar with the center transversal:

▶ **Lemma 3.8.** *Let μ_1 and μ_2 be two finite Borel measures on $\mathbb{R}^d$, which can be separated by a hyperplane. Then there is a unique canonical choice of a center transversal.*

For a proof we refer to the appendix. Again, the same statement holds for point sets in general position. With these tools at hand, we are now ready to prove the second part of the induction.

▶ **Lemma 3.9.** $E(d-1) \Rightarrow R(d)$.

Sketch of Proof. Let ℓ be a line through the origin. Sweep a hyperplane orthogonal to ℓ from one side to the other (without loss of generality from left to right). Let h_1 (h_2) be a sweep hyperplane with exactly $\frac{|R|}{3}$ blue points to the left (right), and let A_1 (A_2) be the set of these blue points. Let M be the set of red points between h_1 and h_2. Note that $|M| \geq \frac{|R|}{3}$. Let c be the unique center transversal of A_1 and A_2 given by Lemma 3.8. By Lemma 3.7, there exists a choice of ℓ, such that c is also a center transversal for M. The projection of c to an orthogonal hyperplane is a centerpoint of the projection of A_1, thus by the statement $E(d-1)$ there are three subsets $A_{1,1}, \dots A_{1,d}$ of A_1, each of size $c_1 \cdot |A_1|$ whose projections enclose the projection of c. Analogously we get subsets $A_{2,1}, \dots, A_{2,d}$ of A_2 and $M_1, \dots, M_d$ of M. By Theorem 3.5 there are subsets $A'_{1,1}, \dots, M'_d$, each of size linear in the size of the original subset, such that each transversal has the same order type. Pick one such transversal. It can be shown that he convex hulls of the blue points (from A_1 and A_2) and the red points (from M) intersect. In particular, there is a subset of $d+2$ red and blue points, which form a Radon partition. By choosing the subsets from which these points were selected, we now get the subsets required for $R(d)$. ◀

4 Conclusion

We have introduced two families of depth measures, called super-additive depth measures and central depth measures, where the first is a strict subset of the second. We have shown that all these depth measures are a constant-factor approximation of Tukey depth.

It is known that Tukey depth is coNP-hard to compute when both $|S|$ and d is part of the input [8], and it is even hard to approximate [2] (see also [5]). Our result is thus an indication that central depth measures are hard to compute. However, this does not follow directly, as

our constant has an exponential dependence on d. It is an interesting open problem whether the approximation factor can be improved.

There is a depth measure which has attracted a lot of research, which does not fit into our framework: simplicial depth (SD). The reason for this is that while the depth studied in this paper are linear in the size of the point set, simplicial depth has values of size $O(|S|^{d+1})$. However, after the right normalization, simplicial depth can be reformulated to satisfy all conditions except super-additivity and centrality. It would be interesting to see whether there is some function g depending on point sets and query points such that the depth measure $\frac{\mathrm{SD}(S,q)}{g(S,q)}$ is super-additive. Such a function, if it exists, could potentially be used to improve bounds for the first selection lemma (see e.g. [11]).

Acknowledgments. Thanks to Emo Welzl, Karim Adiprasito and Uli Wagner for the helpful discussions.

References

1 Greg Aloupis. Geometric measures of data depth. In *Data Depth: Robust Multivariate Analysis, Computational Geometry and Applications*, pages 147–158, 2003.

2 Edoardo Amaldi and Viggo Kann. The complexity and approximability of finding maximum feasible subsystems of linear relations. *Theoretical Computer Science*, 147(1):181 – 210, 1995.

3 Imre Bárány and Pablo Soberón. Tverberg's theorem is 50 years old: a survey. *Bulletin of the American Mathematical Society*, 55(4):459–492, 2018.

4 Imre Bárány and Pavel Valtr. A positive fraction Erdös-Szekeres theorem. *Discrete & Computational Geometry*, 19(3):335–342, 1998.

5 Dan Chen, Pat Morin, and Uli Wagner. Absolute approximation of Tukey depth: Theory and experiments. *Computational Geometry*, 46(5):566 – 573, 2013. Geometry and Optimization.

6 VL Dol'nikov. Transversals of families of sets in in $\mathbb{R}^n$ and a connection between the Helly and Borsuk theorems. *Russian Academy of Sciences. Sbornik Mathematics*, 79(1):93, 1994.

7 Sariel Har-Peled and Timothy Zhou. Improved approximation algorithms for tverberg partitions. *arXiv preprint arXiv:2007.08717*, 2020.

8 D.S. Johnson and F.P. Preparata. The densest hemisphere problem. *Theoretical Computer Science*, 6(1):93 – 107, 1978.

9 Regina Y. Liu, Jesse M. Parelius, and Kesar Singh. Multivariate analysis by data depth: descriptive statistics, graphics and inference. *Ann. Statist.*, 27(3):783–858, 06 1999. URL: `http://dx.doi.org/10.1214/aos/1018031260`, `doi:10.1214/aos/1018031260`.

10 Jiří Matoušek. *Using the Borsuk-Ulam Theorem: Lectures on Topological Methods in Combinatorics and Geometry*. Springer Publishing Company, Incorporated, 2007.

11 Jivr'i Matouvsek. *Lectures on discrete geometry*, volume 212 of *Graduate texts in mathematics*. Springer, 2002.

12 Karl Mosler. *Depth Statistics*, pages 17–34. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. URL: `http://dx.doi.org/10.1007/978-3-642-35494-6_2`, `doi:10.1007/978-3-642-35494-6_2`.

13 Richard Rado. A theorem on general measure. *Journal of the London Mathematical Society*, 21:291–300, 1947.

14 Csaba D Toth, Joseph O'Rourke, and Jacob E Goodman. *Handbook of discrete and computational geometry*. Chapman and Hall/CRC, 2017.

15 John W. Tukey. Mathematics and the picturing of data. In *Proc. International Congress of Mathematicians*, pages 523–531, 1975.

16 Helge Tverberg. A generalization of Radon's theorem. *Journal of the London Mathematical Society*, 1(1):123–128, 1966.

17 Rade T. Zivaljević and Siniša T Vrećica. An extension of the ham sandwich theorem. *Bulletin of the London Mathematical Society*, 22(2):183–186, 1990.

Geometric Dominating Sets - A Minimum Version of the No-Three-In-Line Problem

Oswin Aichholzer[1], David Eppstein[2], and Eva-Maria Hainzl[3]

1 **Graz University of Technology**
`oaich@ist.tugraz.at`
2 **University of California, Irvine**
`eppstein@uci.edu`
3 **Vienna University of Technology**
`eva-maria.hainzl@tuwien.ac.at`

Abstract

We consider a minimizing variant of the well-known *No-Three-In-Line Problem*, the *Geometric Dominating Set Problem*: What is the smallest number of points in an $n \times n$ grid such that every grid point lies on a common line with two of the points in the set? We show a lower bound of $\Omega(n^{2/3})$ points and provide a constructive upper bound of size $2\lceil n/2 \rceil$. If the points of the dominating sets are required to be in general position we provide optimal solutions for grids of size up to 12×12. For arbitrary n the currently best upper bound remains the obvious $2n$. Finally, we discuss some further variations of the problem.

1 Introduction

The well-known *No-Three-In-Line Problem* asks for the largest point set in an $n \times n$ grid without three points in a line. This problem has intrigued many mathematicians including e.g. Paul Erdős for roughly 100 years now. Few results are known and explicit solutions obtaining the trivial upper bound of $2n$ only exist for n up to 46 and $n = 48, 50, 52$ (See e.g. [3]). Providing general bounds seems to be notoriously hard to solve; see [4, 6] for some history of this problem.

In this note we concentrate on an interesting minimizing variant of the No-Three-In-Line problem, which we call the *Geometric Dominating Set Problem*: What is the smallest number of points (or points in general position) in an $n \times n$ grid such that every grid point lies on a common line with two of the points in the set? This problem arose during the 2018 Bellairs Winter Workshop on Computational Geometry. Later we found out that already in 1976 in Martin Gardner's Mathematical Games column [4] the minimization version has been mentioned. Gardner wrote: "*Instead of asking for the maximum number of counters that can be put on an order-n board, no three in line, let us ask for the minimum that can be placed such that adding one more counter on any vacant cell will produce three in line.*" According to Gardner, the problem had already been mentioned briefly in a paper by Adena, Holton and Kelly [1]. He mentioned their best results which they obtained by hand for $3 \leq n \leq 10$. These are $4, 4, 6, 6, 8, 8, 12, 12$. Surprisingly, up to $n = 8$, their solutions are indeed optimal solutions as we will see in Section 3. However, it seems that no progress has been made since then, except for the special case where lines are restricted to vertical, horizontal and 45° diagonal lines [2].

This minimum version might remind one less of the No-Three-In-Line Problem, which itself is based on a mathematical chess puzzle, and more of the *Queens Domination Problem* that asks for a placement of five queens on a chessboard such that every square of the board is attacked by a queen. In a more general setting this problem asks for the domination

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

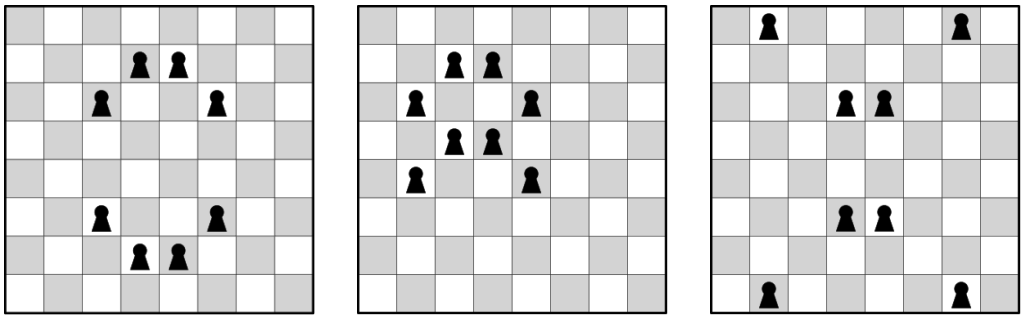


Figure 1 Three out of 228 solutions: Every square lies on a line defined by two pawns where no three pawns are allowed to lie on a common line.

number of the $n \times n$ queen graph. Inspired by that, we call the smallest size of a solution for the Geometric Dominating Set Problem the *geometric domination number* $\mathcal{D}_n$.

After introducing the Geometric Dominating Set Problem formally, we will prove non-trivial asymptotic upper and lower bounds and provide further computational results.

1.1 Dominating Sets

In the spirit of mathematical chess puzzles, the Geometric Dominating Set Problem can be formulated in two variants as

How many pawns do we have to place on a chessboard such that every square lies on a straight line defined by two pawns? How many pawns do we need if no three pawns are allowed to lie on a common line?

We will see in Section 3, the answer for a chessboard is eight and some solutions are the placements shown in Figure 1. In fact, there are 228 possibilities to do so, and 44 if we cancel out rotation and reflection symmetries.

▶ **Definition 1.1.** Three points are called *collinear*, if they lie on a common line. Conversely, a set S is called *in general position* if no three points in S are collinear.

We call a point p in the $n \times n$ grid *dominated (by a set S)*, if $\mathsf{p} \in S$ or there exist $\mathsf{x}, \mathsf{y} \in S$ such that $\{\mathsf{x}, \mathsf{y}, \mathsf{p}\}$ are collinear. Similarly, we say p is dominated by a line L if p lies on L.

A subset S of the $n \times n$ grid is called a *(geometric) dominating set* or simply *dominating* if every point in the grid is dominated by S.

We call the smallest size of a dominating set of the $n \times n$ grid the *(geometric) domination number* and denote it by $\mathcal{D}_n$. The smallest size of a dominating set in general position (an *independent dominating set*) is called the *independent (geometric) domination number* and denoted by $\mathcal{I}_n$. Note that every point in an independent dominating set is only dominated by pairs that include the point itself.

1.2 Summary Of Results

We will show that

- $\mathcal{I}_n \geq \mathcal{D}_n = \Omega(n^{2/3})$ (Subsection 2, Theorem 2.3)
- $\mathcal{D}_n \leq 2\lceil n/2 \rceil$ (Subsection 3, Theorem 3.1)

and present several computational results.

2 Lower Bounds

For a lower bound on $\mathcal{D}_n$, let us consider a set S of s points in the $n \times n$ grid. Any pair of points in S can dominate at most n points, so it has to hold that $\binom{s}{2}n \geq n^2$ which is the case if $s \geq \frac{1}{2} + \sqrt{\frac{1}{4} + 2n} \geq \sqrt{2n}$. Therefore, $\mathcal{D}_n = \Omega(n^{1/2})$.

However, hardly any lines in the $n \times n$ grid dominate n points. In fact, we can prove a significantly better bound by using the following theorem, where φ denotes the Euler-Phi function.

▶ **Theorem 2.1.** *Let $n = 2k + 1$ and S be a subset of the $n \times n$ grid with $|S| \leq 4\sum_{i=1}^{m} \varphi(i)$, where $1 \leq m \leq k$. Then the number of points dominated by lines incident to a fixed point $\mathsf{x} \in S$ and the other points in S is bounded by*

$$1 + 8\sum_{i=1}^{m} \left\lfloor \frac{n}{i} \right\rfloor \varphi(i) \;\leq\; \frac{48}{\pi^2} nm + O\left(n \log m\right).$$

The proof of this theorem can be found in [6] and uses the following well known number theoretic result.

▶ **Theorem 2.2** (Arnold Walfisz [7])**.**

$$\sum_{i=1}^{k} \varphi(i) = \frac{3}{\pi^2} k^2 + O\left(k(\log k)^{\frac{2}{3}} (\log\log k)^{\frac{4}{3}}\right)$$

$$\sum_{i=1}^{m} \frac{\varphi(i)}{i} = \frac{6}{\pi^2} m + O\left((\log m)^{\frac{2}{3}} (\log\log m)^{\frac{4}{3}}\right)$$

▶ **Theorem 2.3** (A lower bound on $\mathcal{D}_n$)**.** *For $n \in \mathbb{N}$, it holds that $\mathcal{D}_n = \Omega(n^{2/3})$.*

Proof. First, let $n = 2k + 1$, $k \in \mathbb{N}$ and let S be a set of s points in the grid, where $\sqrt{2n} \leq s \leq 2n$. (Recall that $2n$ is a trivial upper bound on $\mathcal{D}_n$ and $\sqrt{2n}$ a lower bound.) Let m be the smallest positive integer such that $s \leq 4 \cdot \sum_{i=1}^{m} \varphi(i)$. Then $s \sim \frac{12}{\pi^2} m^2$ by Theorem 2.2.

By Theorem 2.1, the number of points dominated by lines incident to a fixed point p and one of $s - 1$ additional points is bounded by $\frac{48}{\pi^2} nm + O\left(n \log m\right)$. To dominate all points in the grid, we thus need

$$n^2 \leq s\left(\frac{48}{\pi^2} nm + O\left(n \log m\right)\right).$$

Next, we plug in the asymptotic expression for s, such that the inequality simplifies to

$$n^2 \leq \left(\frac{12}{\pi^2} m^2 + O\left(m \log m\right)\right)\left(\frac{48}{\pi^2} nm + O\left(n \log m\right)\right) = \frac{576}{\pi^4} nm^3 + O\left(nm^2 \log m\right)$$

If we divide by n, we can see that $m = \Omega(n^{1/3})$ and consequently $s = \Omega(n^{2/3})$ which proves the claim for n odd.

For n even we embed the $n \times n$ grid into the $(n + 1) \times (n + 1)$ grid and obtain the same asymptotic results. ◀

▶ **Corollary 2.4.** $\mathcal{I}_n = \Omega(n^{2/3})$.

Proof. Since any independent dominating set is a dominating set, $\mathcal{I}_n \geq \mathcal{D}_n$. ◀

3 Upper Bounds

▶ **Theorem 3.1** (An upper bound on $\mathcal{D}_n$). *For $n \in \mathbb{N}$, it holds that $\mathcal{D}_n \leq 2\left\lceil \frac{n}{2} \right\rceil$.*

The proof is based on the construction in Figure 2 and can be found in [6]. If $n = k^2$ is an odd square the result can be slightly improved to $n - 1$ by a construction similar to the one depicted for $k = 3$ and $n = 9$ in the leftmost drawing of Figure 6.

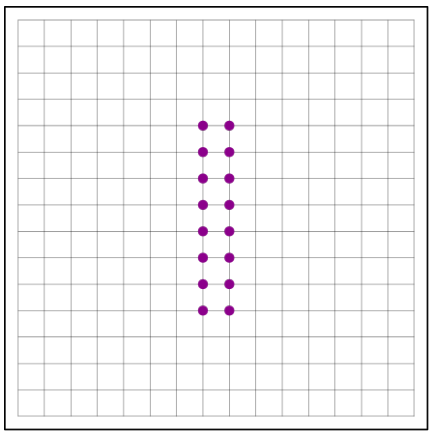


Figure 2 Dominating set construction for $n = 16$.

So far, for $\mathcal{I}_n$ there is no better upper bound known than the obvious $2n$.

4 Small Cardinalities and Examples

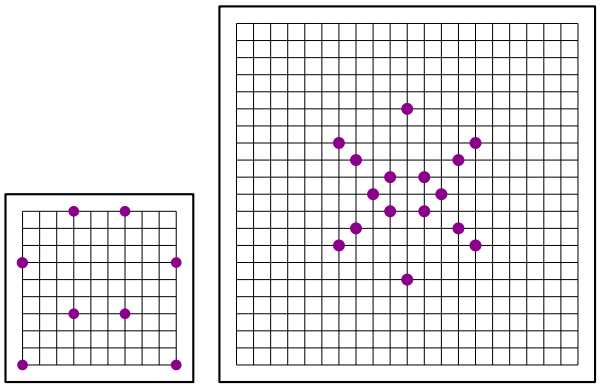


Figure 3 The unique (up to symmetry) minimal independent dominating set of size 8 for the 10×10 board and a small independent dominating set of size 16 for the 21×21 board. The latter gives the currently best known ratio (number of points / grid size) of $16/21 < 0,762$.

To obtain results for small grids we developed a search algorithm based on the classic backtracking approach. To speed up the computation, both symmetries – rotation and reflection – were taken into account. For $n = 2, \ldots, 12$, we made an exhaustive enumeration of all independent dominating sets, and the obtained results are summarized in Table 1. For larger sets upper bounds on $\mathcal{I}_n$ are given in Table 2. Figure 3 gives two examples of small independent dominating sets.

n	2	3	4	5	6	7	8	9	10	11	12
$\mathfrak{I}_n$	4	4	4	6	6	8	8	8	8	10	10
non sym. sets	1	2	2	26	2	573	44	3	1	19	2
all sets	1	5	2	152	8	4136	228	11	4	108	12

Table 1 Size of smallest independent dominating sets for $n = 2, \ldots, 12$ and number of different sets, considering symmetry (rotation and reflection), and all sets.

n	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
$\mathfrak{I}_n \leq$	12	12	14	14	15	16	16	16	16	18	20	20	22	24	24	24	24	25

Table 2 Currently best upper bounds for smallest independent dominating sets for $n > 12$.

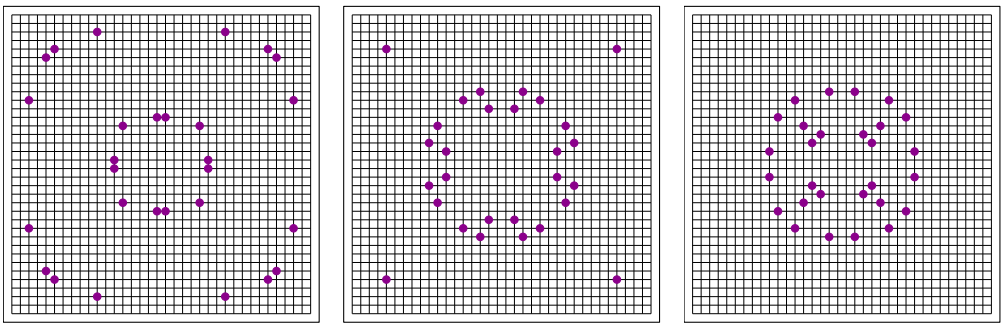


Figure 4 Three independent dominating sets of cardinality 28 for a 36×36 board.

We also obtained results for larger sets, but there is no evidence that our sets are (near the) optimal solutions. Most of these examples are rather symmetric, but that might be biased due to the approach we used to generate larger sets from smaller sets by adding symmetric groups of points. Figure 4 shows three drawings for $n = 36$ with independent dominating sets of size 28.

Figure 5 shows different dominating sets for $n = 7$. The best dominating sets that contain collinear points are smaller than the best solutions in general position. For $n \leq 12$ this is the only board size where allowing collinear points leads to smaller dominating sets. Figure 6(left) shows some nicely symmetric dominating sets with collinear points.

5 Variations of the Problem and Conclusion

We have already seen in the previous section that minimal examples can get smaller if we allow dominating sets to contain collinearities, cf. Figures 5 and 6. We can also release the restriction that the points of the dominating set have to lie within the grid, that is, the points can have coordinates smaller than one, or larger than n. In Figure 6(right) we depict two examples where the shown dominating sets are smaller than the best bounded solutions in general position. So far we have not been able to find any examples where this idea combined with collinear points in the dominating set provided even better solutions.

Another interesting variant is a game version: Two players alternatingly place a point on the $n \times n$-grid such that no three points are collinear. The last player who can place a valid point wins the game (called normal play in game theory). It is not hard to see that for

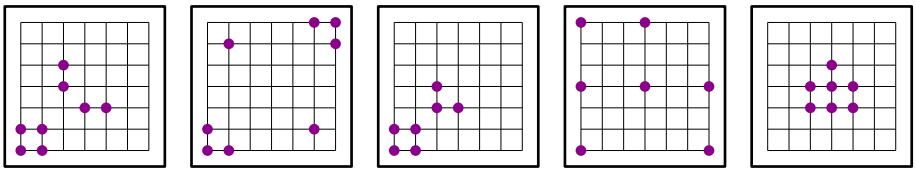


Figure 5 Five different dominating sets for a 7×7-board. The first two sets are in general position and have size 8, while the remaining three sets have size 7 but contain collinear points.

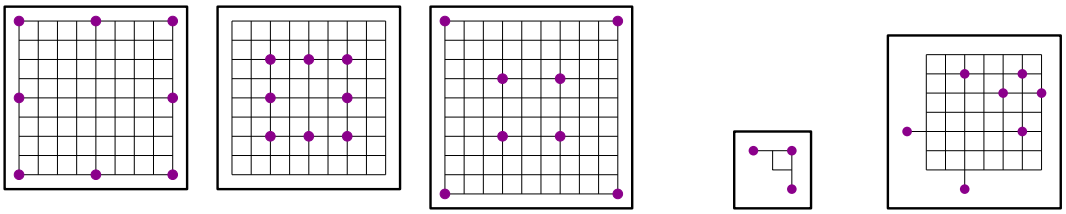


Figure 6 Left: Symmetric dominating sets with collinear points for $n = 9$ and $n = 10$. Right: Smaller dominating sets for a 2×2-board and a 7×7-board if points are allowed to be outside of the board. These solutions are unique up to symmetry.

any even n the second player has a winning strategy. She just always sets the point which is center mirrored to the previous move of the first player. By symmetry arguments this move is always valid, as long as the first player made a valid move. For n odd the situation is more involved. If the first player does not start by placing the central point in her first move, then we have again a winning strategy for the second player by the same reasoning (note that the central place can not be used after the first two points have been placed, as this would cause collinearity). So if the first player starts by placing the central point it can be shown that for $n = 3$ she can also win the game. But for $n = 5, 7, 9$ still the second player has a winning strategy. For odd n we currently do not know the outcome for games on grids of size $n \geq 11$.

Several open problems arise from our considerations:

- Is there a constant $c > 0$ such that $\mathcal{D}_n \leq \mathcal{I}_n \leq (2 - c)n$ holds for large enough n?
- Do $\mathcal{I}_n$ and $\mathcal{D}_n$ grow in a monotone way, that is, is $\mathcal{I}_{n+1} \geq \mathcal{I}_n$ and $\mathcal{D}_{n+1} \geq \mathcal{D}_n$?
- Is there some n_0 such that $\mathcal{I}_n > \mathcal{D}_n$ for all $n \geq n_0$?
- Do minimal dominating sets in general position always have even cardinality? For $n \leq 12$ this is the case, but the currently best example for $n = 17$ might be a counterexample.
- How much can the size of dominating sets (with or without collinear points) be improved if the points are allowed to lie outside the grid?
- Which player has a winning strategy in the game version for boards of size $n \geq 11$, n odd.

In [6], the problem was also considered on the discrete $n \times n$ torus. By extending the probabilistic approach of Guy and Kelly to the No-3-In-Line problem [5] an upper bound of $O(\sqrt{n \log n})$ holds, which remarkably is below the lower bound of the regular grid. We can show a lower bound of $\Omega(\sqrt{n})$ for the torus if n is prime, but if n is a power of 2, then actually 4 points are sufficient. We will provide detailed results in the full version.

Acknowledgments. This research was initiated at the 33^{rd} Bellairs Winter Workshop on Computational Geometry in 2018, and continued at the 2019 edition of this workshop. We thank the organizers and participants of both workshops for a very fruitful atmosphere.

References

1 M.A. Adena, D.A. Holton, and P.A. Kelly. Some thoughts on the no-three-in-line problem. In J. Seberry, editor, *Combinatorial Mathematics: Proceedings of the Second Australian Conference, August 16-27, 1977*, volume 403 of *Lecture Notes in Mathematics*, pages 6–17. Springer, 1974. doi:10.1007/BFb0057371.

2 A.S. Cooper, O. Pikhurko, J.R. Schmitt, and G.S. Warrington. Martin Gardner's minimum no-3-in-a-line problem. *American Mathematical Monthly*, 121(3):213–221, 2014. doi:10.4169/amer.math.monthly.121.03.213.

3 A. Flammenkamp. Solutions to the no-three-in-line problem. Available on his homepage at the University of Bielelefeld, 1997. Retrieved on April 29, 2020. URL: http://wwwhomes.uni-bielefeld.de/achim/no3in/table.txt.

4 M. Gardner. Mathematical games: Combinatorial problems, some old, some new and all newly attacked by computer. *Scientific American*, 235(4):131–137, October 1976. URL: https://www.jstor.org/stable/24950467.

5 R. Guy and P. Kelly. The no-three-in-line problem. *Canadian Mathematical Bulletin*, 11:527–531, 1968. doi:10.4153/CMB-1968-062-3.

6 E.M. Hainzl. Geometric dominating sets. Master's thesis, TU Graz, July 2020.

7 A. Walfisz. *Weylsche Exponentialsummen in der neueren Zahlentheorie*. Deutscher Verlag der Wissenschaften, Berlin, 1963.

Polyline Bundle Simplification on Trees

Yannick Bosch, Peter Schäfer and Sabine Storandt

University of Konstanz
{yannick.bosch, peter.schaefer, sabine.storandt}@uni-konstanz.de

Abstract

The polyline bundle simplification problem asks for the smallest consistent simplification (with respect to a given distance threshold) of a set of polylines which may share line segments and bend points. As the problem was shown to be APX-hard in previous work, we consider here an interesting special case where the polylines form a rooted tree. Tree bundles naturally arise in the context of movement data visualization. Moreover, general bundles might be decomposable into a (small) set of tree bundles. We present an algorithm that computes optimal tree bundle simplifications in time $\mathcal{O}(n^3)$ where n is the total number of points in the input. The applicability of our approach is demonstrated in an experimental evaluation on real-world data.

1 Introduction

Polyline simplification is a well-studied optimization problem which can be solved to optimality in polynomial time [3]. However, in case the input is a set of (partially) overlapping polylines, individual simplification of each polyline leads to visually unpleasing results as shared parts may be simplified in different ways. Aiming at more appealing results, Spoerhase et al. [5] introduced the problem of *Polyline Bundle Simplification (PBS)*, adding as an additional constraint that shared parts must be simplified consistently (i.e. each point is either kept in or discarded from all polylines containing it).

▶ **Definition 1** (Polyline Bundle Simplification [5]). An instance of PBS consists of a triple $(P, \mathcal{L}, \delta)$ where $P = (p_1, \ldots, p_n)$ is a set of n points in the plane, $\mathcal{L} = \{L_1, \ldots L_l\}$ is a set of simple polylines, each represented as lists of points from P, and δ is a distance parameter. Given a triple $(P, \mathcal{L}, \delta)$, the goal is to obtain a minimum size subset $P^* \subseteq P$ of points such that for each polyline $L \in \mathcal{L}$ its induced simplification $S = L \cap P^*$ contains the start and end point of L and has a segment-wise Fréchet distance of at most δ to L.

PBS is a generalization of the classical polyline simplification problem but was proven by Spoerhase et al. [5] to be NP-hard to approximate within a factor of $n^{1/3-\varepsilon}$ for any $\varepsilon > 0$ even for two polylines. As a potentially practical feasible approach, a bi-criteria $(O(\log(l+n)), 2)$-approximation algorithm was presented in [5]. This algorithm is allowed to return results within a distance threshold of 2δ, and based on this constraint relaxation achieves a logarithmic approximation factor (compared to the optimal solution for δ) in polynomial time.

With the general problem being APX-hard, we focus in this paper on designing efficient algorithms for a special case of PBS in which the polylines form a rooted tree. Such polyline bundles arise for example when aggregating vehicle trajectories from a certain starting location, with the vehicles moving on (unique) shortest paths in an underlying road network. Similar to the concept of the Imai-Iri algorithm for simplification of a single polyline [3], our algorithm precomputes the possible set of shortcuts for the given distance threshold and thereupon transforms the given geometric problem into a graph problem. But while in the Imai-Iri algorithm a simple search for the minimum link-path in the shortcut graph suffices, we need a more intricate dynamic programming approach to deal with the tree structure.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

2 Problem Statement

We now formally define the concept of a *Polyline Tree Bundle (PTB)*.

▶ **Definition 2** (Polyline Tree Bundle)**.** A PTB is a set of simple polylines $\mathcal{L}$ where all $L \in \mathcal{L}$ start at the same root point p_r, and for any pair of polylines $L, L' \in \mathcal{L}$ the only allowed intersection is a common prefix $(p_r, ..., p_i)$.

As described in Definition 1 for general polyline bundles, our optimization goal is to find for a given distance threshold δ the smallest subset of points in the input such that the induced simplification is valid with respect to δ for each $L \in \mathcal{L}$. A *distance function* d is used to measure the distance of a line segment (a, b) in the simplification to the corresponding sub-polyline of L, which we abbreviate by $L[a, \ldots, b]$. For a valid simplification, we require $d((a, b), L[a, \ldots, b]) \leq \delta$. In the following, we will consider two distance functions: (i) the Fréchet distance (as used in [5]), and (ii) the maximum Euclidean distance.

To transform the PTB simplification problem into a graph problem, we construct two directed graphs from the input data: a *tree graph* and a *shortcut graph*. We start by considering the polylines as embedded directed paths which start at the root point. The tree graph $G_T = (V, E_T)$ is the union of these paths. More precisely, for each point p occurring in the PTB there is a corresponding node $v \in V$ (with v_r corresponding to the root point p_r), and there exists a directed edge $(v, w) \in E_T$ if there is a polyline $L \in \mathcal{L}$ which contains the segment between the respective points (in that direction). For a given distance function d and threshold $\delta > 0$, the shortcut graph $G_S = (V, E_S)$ is the union of all valid shortcut edges, i.e. edges $(v, w) \in \binom{V}{2}$ where for all polylines $L \in \mathcal{L}$ that contain v and w (in that order), we have $d((v, w), L[v, \ldots, w]) \leq \delta$. Note that $E_T \subseteq E_S$, i. e. all tree graph edges are also contained in the shortcut graph.

The construction of these graphs can be accomplished in time $\mathcal{O}(n^3)$. In a PTB on n points, there can be at most n polylines with a total of $\mathcal{O}(n^2)$ segments. Accordingly, the tree graph can be constructed in time $\mathcal{O}(n^2)$. We remark that we do not need to consider the case where a polyline $L' \in \mathcal{L}$ is a sub-polyline of $L \in \mathcal{L}$. By definition, we will include the endpoints of all polylines in our simplification and hence if the endpoint of L' lies on L, we could simply consider that point as the root of another PTB which can be simplified independently. For the shortcut graph construction, we compute the set of valid shortcuts by considering the polylines one after the other. To avoid redundant computations along shared parts, we store the result for already considered node pairs. Accordingly, the total number of potential shortcuts that need to be checked is in $\mathcal{O}(n^2)$. The time T_d to check the validity of a shortcut depends on the distance function d. For both, the Fréchet distance and the Euclidean distance, we have $T_d \in O(n)$. Therefore, the total construction time is in $\mathcal{O}(n^3)$.

We are now ready to restate the PTB simplification problem (PTBS) as a graph problem.

▶ **Definition 3** (Polyline Tree Bundle Simplification)**.** Given a tree graph $G_T(V, E_T)$ and a shortcut graph $G_S(V, E_S)$, the goal is to find a smallest node subset $S \in V$ such that:

- The root node and all leaf nodes of the tree graph are contained in S.
- For each path from the root node to a leaf node in G_T, its restriction to nodes in S has to result in a valid path in G_S (i.e. consecutive nodes are connected with a valid shortcut).

In Figure 1, optimal PTBS solutions for an example instance are shown.

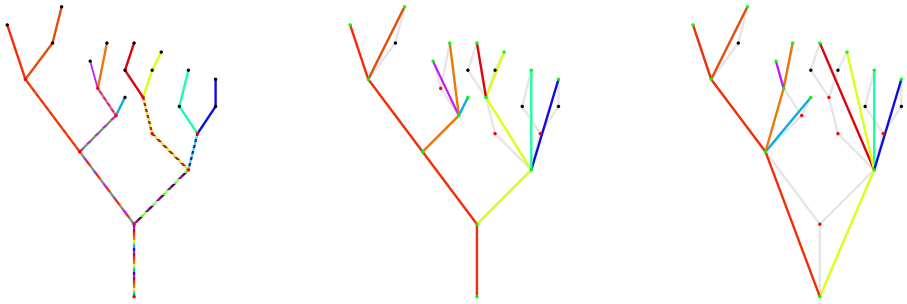


Figure 1 A tree bundle and two optimal consistent simplifications for different distance thresholds.

3 A Polynomial Time Algorithm for PTBS

In this section, we describe a dynamic programming approach that operates on the tree graph G_T and the shortcut graph G_S, and returns an optimal PTBS solution in $\mathcal{O}(n^2)$.

Let $Sub(v) \subseteq G_T$ be the sub-tree rooted at node v in the tree graph. Our main observation is that we can break down an optimal solution recursively. If a node v is part of the solution ($v \in S$), it's easy to see that there can't be shortcuts bypassing v. Thus, the solution S can be split into two parts: an optimal solution for $Sub(v)$ and an optimal solution for $G_T \setminus Sub(v)$. We denote the size of an optimal solution for $Sub(v)$ by $s(v)$. As we don't know a priori which nodes will end up in the solution, we strive for computing $s(v)$ for each node $v \in V$ in an efficient manner. For leaf nodes v, we obviously get $s(v) = 1$. To compute $s(v)$ for an inner node v, we assume that $s(w)$ is already known for all nodes $w \in Sub(v) \setminus \{v\}$. For each path from v to a leaf node in $Sub(v)$, the solution for $Sub(v)$ needs to contain a cover node w such that $(v, w) \in E_S$ (that means there is a valid shortcut from v to w). To identify the best selection of cover nodes, we compute a helping function $h : V \to \mathbb{N}$ for each node $w \in Sub(v)$ as follows: Initially, $h(w) = s(w)$ if $(v, w) \in E_S$, and $h(w) = \infty$ otherwise. Then, in a post-order traversal of $Sub(v)$, we set $h(w) = \min\{h(w), \sum_{u \in N(w)} h(u)\}$ where $N(w)$ denotes the set of children (out-neighbors) of w in G_T. In that way, $h(w)$ encodes the smallest number of nodes that have to be kept in $Sub(w)$ if for all paths from v to leaf nodes in $Sub(w)$ the respective cover node is contained in $Sub(w)$. The optimal solution size $s(v)$ for $Sub(v)$ is then equal to $h(v) + 1$ (as we have to additionally include v itself). Note that $s(v)$ is always well-defined (i.e., finite) as the tree edges are all valid shortcuts in G_S. The time to compute $s(v)$ is in $\mathcal{O}(|Sub(v)| + |\{(v, w) \in E_s\}|)$ and hence can be upper bounded by $\mathcal{O}(n)$. To make sure that at the time we want to compute $s(v)$ all values $s(w)$ for $w \in Sub(v) \setminus \{v\}$ are known, we also globally traverse the nodes in the tree graph in post-order. Hence altogether, we have two nested post-order traversals with a total complexity of $O(n^2)$. The optimal set of simplification nodes S can then be determined by backtracking.

For a faster running time in practice, we suggest to only compute h-values for nodes in $Sub(v)$ which are on a path from v to some potential cover node w with $(v, w) \in E_S$. These nodes can easily be identified by computing the reverse path from each such node w to v and marking all nodes along the way (stopping as soon as a marked node is encountered to avoid redundancy). For marked nodes w with an unmarked neighbor, we simply set $\sum_{u \in N(w)} h(u)$ to ∞ to maintain correctness. Especially for small distance thresholds δ and large sub-trees $Sub(v)$, this modification is expected to accelerate the computation of $s(v)$ significantly.

Finally, we remark that minimizing the number of nodes in the simplified tree bundle is equivalent to minimizing the number of segments; while for general polyline bundles these two optimization goals might lead to different results as discussed in [5].

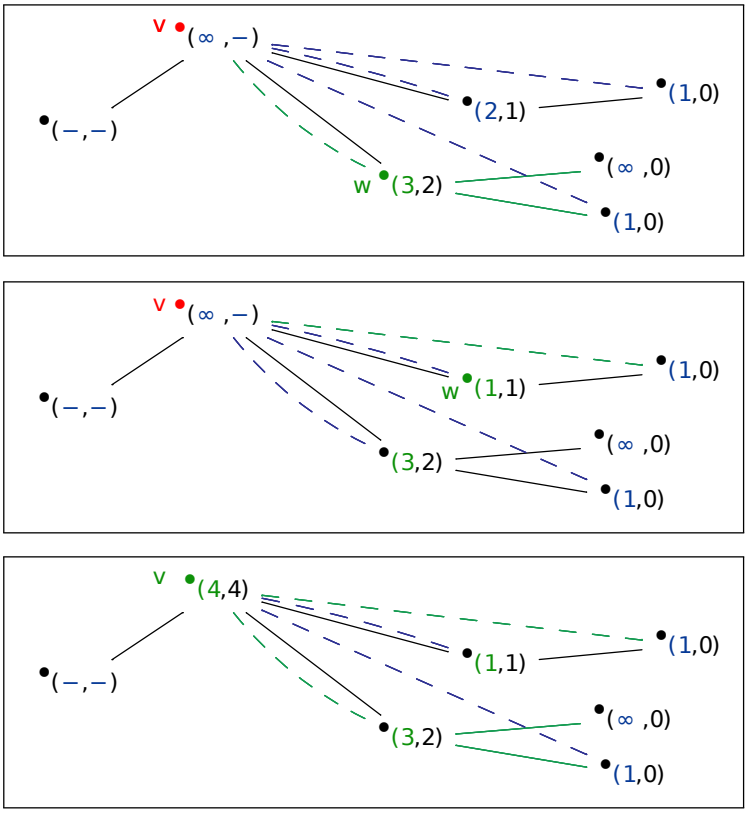


Figure 2 Example: computation of $h(w)$. Tree egdes are black, shortcuts are dashed blue, solution shortcuts are dashed green. Attached to each node: $h(w)$ and $s(w)$.

In the first figure, w is currently reachable by a shortcut, so is one of its children. However, since one child is not reachable by a shortcut (indicated by $h = \infty$), we compute $h(w) = s(w) + 1$.

In the second figure, $h(w)$ is updated to 1, because the only child is reachable by a shortcut, and the sum of h over the children is smaller than the original $h(w)$.

In the bottom figure, we compute $s(v) = h(v)$ as the sum of h over all children.

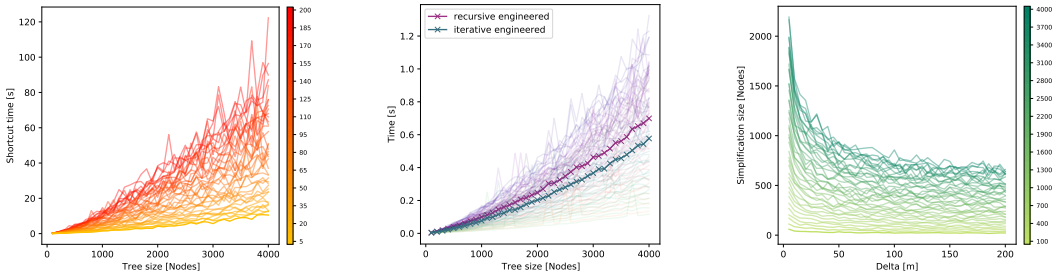


Figure 3 Experimental results on tree bundles of different size extracted from road networks. The left image shows the time (in seconds) to compute the shortcut graph in dependency of the number of nodes in the input. More orange lines indicate larger values of δ. The image in the middle provides running time results of the DP approach (subdivided by different means to compute a post-order of the nodes in the tree graph). The right image depicts the computed optimal simplification sizes. Here, darker lines indicate a larger number of nodes in the input. Points are averaged over 20 tree bundles for every configuration of δ and the tree size. δ is given in meters.

4 Experiments

We implemented the dynamic programming-based approach for PTBS as well as the bi-criteria approximation algorithm by Spoerhase et al. [5] suited for simplification of general polyline bundles. Our input is constructed by randomly sampling a node from the road network of Germany (extracted from OSM). Expanding from the root we construct a subgraph with a certain size and perform a BFS to obtain a tree. A tree bundle is then generated by backtracking from the leaves. Experiments were mainly conducted on a single core of an AMD Ryzen 7 3700X with 4.1 GHz. Bi-criteria algorithm experiments were conducted on an Intel Core i9 with 2.4 GHz.

Figure 3 illustrates the experimental results for our proposed approach on tree bundles of different size and with different distance thresholds δ (limiting the Euclidean distance). In compliance with our theoretical analysis, we observe that the time to compute the shortcut graph clearly dominates the overall running time. The dynamic program (DP) itself then produces the optimal solution very quickly, especially when using the engineering idea described above and an iterative depth-first-search run in the tree to compute a post-order of the nodes. In the right plot in Figure 3, it can be nicely observed that the optimal number of nodes in the simplification converges for growing values of δ to the number of leaf nodes in the tree graph, as those have to be kept by definition.

Furthermore, we compared the performance of the DP-based approach for PTBS to that of the bi-criteria approximation algorithm (BCA) on the same input with the Fréchet distance serving as the distance function. This experiment was conducted on a single core of an Intel i5-4300U CPU with 1.90GHz and 12GB RAM. As BCA might return results where the simplified polylines have a segment-wise Fréchet distance of up to 2δ to the original polylines, we ran the BCA algorithm twice, once with δ and once with $\delta/2$. Table 1 shows the respective results for the example tree bundle illustrated in Figure 4 and $\delta = 0.00005$.

We observe that BCA indeed makes use of the relaxed distance constraint. In the BCA run with $\delta = 0.00005$, the resulting Fréchet distance is 1.75 times larger than δ, coming close to the theoretically guaranteed upper bound of 2. However, even with this increased distance threshold, the BCA solution is worse than the optimal solution for the original δ computed with the DP-based approach. And if we use BCA with $\delta/2$ to ensure that it enforces the same distance bound as the DP-approach, the quality deteriorates further. In

Figure 4 Polyline tree bundle (based on the road network of Konstanz) with 8000 points.

addition, the DP-based approach is about 20 times faster. The same trend was observed on other instances. We hence conclude that on tree bundles the DP-based approach is superior. But of course BCA is more versatile as it can be applied to arbitrary polyline bundles.

	δ	Fréchet distance d_F	d_F/δ	retained points	time (msec)
DP	0.00005	0.0000500	1.00	4009	21
BCA	0.00005	0.0000877	1.75	4029	407
BCA	0.000025	0.0000404	1.61	5276	350

Table 1 Comparison of the DP and the BCA algorithm on an input tree bundle with 8000 points. Distances are given in geo-coordinates (0.00005 decimal degrees $\approxeq 5m$).

5 Outlook

We developed an efficient and exact approach for polyline tree bundle simplification. As shown in our theoretical as well as empirical analysis, the running time is currently dominated by the computation of the shortcut graph. Therefore, attempts to accelerate the approach should focus on this step. For example, a shortcut (v, w) bypassing a node z of out-degree larger than 1 can only be part of a feasible solution if for every subtree rooted at an out-neighbor of z there is also a valid shortcut emerging from v that ends in that subtree. It hence might be possible to reduce the number of potential shortcuts that have to be checked based on structural observations.

Melkman and O'Rourke [4] and Chan and Chin [1] construct a shortcut graph for the Euclidean distance in time $O(n^2)$. We will examine if their results are applicable to our problem setting.

Another interesting direction for future work is the development of techniques to subdivide a general polyline bundle into a set of tree bundles, which then could be simplified optimally (and in parallel) with our approach. If only few points of the general bundle need to be included in the solution set to realize a subdivision into tree bundles, then we would expect to end up with a good overall simplification quality. We assume that real-life movement trajectories might be well suited for such an approach.

References

1 W. Chan and Francis Chin. Approximation of polygonal curves with minimum number of line segments. volume 6, pages 378–387, 1992. doi:10.1007/3-540-56279-6_90.

2 Chenglin Fan, Omrit Filtser, Matthew J. Katz, Tim Wylie, and Binhai Zhu. On the chain pair simplification problem. In *Algorithms and Data Structures*. Springer International Publishing, 2015.

3 Imai Hiroshi and Iri Masao. Polygonal approximations of a curve—formulations and algorithms. In *Machine Intelligence and Pattern Recognition*, volume 6, pages 71–86. Elsevier, 1988.

4 Avraham Melkman and Joseph O'Rourke. On polygonal chain approximation. In Godfried T. Toussaint, editor, *Computational Morphology*, volume 6, pages 87–95. 1988. doi:https://doi.org/10.1016/B978-0-444-70467-2.50012-6.

5 Joachim Spoerhase, Sabine Storandt, and Johannes Zink. Simplification of polyline bundles. In *17th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

Folding Polyiamonds into Octahedra

Eva Bolle[1] **and Linda Kleist**[1]

1 **Technische Universität Braunschweig, Germany**
{eva.bolle,l.kleist}@tu-bs.de

Abstract

We study polyiamonds (triangular polyominoes) that fold into the smallest yet unstudied platonic solid – the octahedron. We present a characterization for the convex polyiamonds that fold into an octahedron. Moreover, we show a sharp size bound for foldable polyiamonds. While there exist unfoldable polyiamonds of size 14, every polyiamond of size at least 15 folds into the octahedron.

1 Introduction

Algorithmic origami is a comparatively young branch of computer science that studies the algorithmic aspects of folding various materials. The construction of three-dimensional objects from two-dimensional raw materials is of particular interest and has applications in robotics in general [10, 12], and also in the construction of objects in space [9].

While foldings of polycubes and tetrahedra have already been studied, we take the next step and focus on the question of whether a given polyiamond folds into the octahedron?

(a) The (unit) octahedron $\mathcal{O}$. **(b)** A polyiamond P.

Figure 1 Does the polyiamond P fold into the octahedron $\mathcal{O}$?

By an *octahedron*, we refer to the regular octahedron $\mathcal{O}$ composed of eight equilateral (unit) triangles; for an illustration consider Figure 1(a). Note that in each of the six vertices four triangles meet. Because all faces of the octahedron are triangles, our pieces of paper are polygons arising from the triangular grid. A *polyiamond* of size n is a connected polygon in the plane formed by joining n triangles from the triangular grid by identifying some of their common sides; for an example consider Figure 1(b). To avoid confusion with the corners of the octahedron, we refer to the vertices of the triangles forming P as the *vertices* of P; note that these vertices may also lie inside P.

We view P as a set which includes the n open triangles and a subset of the shared unit-length boundary edges; the existence of such an edge models the fact that the triangles are glued along this side. Because we only want robust connections between triangles via their sides, we do not specify the existence/non-existence of vertices which do not influence the foldability. However, for the upcoming definitions of slits and holes, we assume that the vertices do not belong to the polyiamond. If a shared edge does not belong to P, we call it a *slit edge*. We also allow the polyiamonds to have holes; a *hole* of a polyiamond is a bounded connected component of its complement, which is different from a single vertex. We call a hole a *slit* if it has area zero and consists of one or more slit edges. We consider

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

two polyiamonds to be *the same* if they are congruent, i.e., if they can be transformed into one another by a set of translations, rotations and reflections. Moreover, a polyiamond is *convex* if it forms a convex set in the plane.

Folding model. We consider foldings in the *grid folding model*, where folds along the grid lines are allowed such that in the final state every triangle covers a face of the octahedron, i.e., we forbid folding material strictly outside or inside the octahedron. Consequently, in the final state the folding angles are $\pm\beta := \arccos(1/3)$ or $\pm 180°$. A folding of a polyiamond P into a (partial) octahedron $\mathcal{O}$ induces a *triangle-face-map*, i.e., a mapping of the triangles of P to the faces of $\mathcal{O}$. We say P *folds into* $\mathcal{O}$ (or also P *is foldable*), if P can be transformed by folds along the grid lines into a folded state such that the induced triangle-face-map is surjective, i.e., each face of $\mathcal{O}$ is covered by at least one triangle.

Related work. Past research has particularly focused on folding polyominoes into *polycubes*. Allowing for folds along the *box-pleat grid* (square grid lines and alternating diagonals), Benbernou et al. (with differing co-authors) show that every polycube Q of size n can be folded from a sufficiently large square polyomino [6] or from a $2n \times 1$ strip-like polyomino [5]. Moreover, common unfoldings of polycubes have been investigated in the grid model. The *(square) grid model* allows folds along the grid lines of a polyomino with fold angles of $\pm 90°$ and $\pm 180°$, and allows material only on the faces of the polyhedron. Benbernou et al. show that there exist polyominoes that fold into all polycubes with bounded surface area [5] and Aloupis et al. study common unfoldings of various classes of polycubes [4]. Moreover, there exist polyominoes that fold into several different boxes [1, 11, 13, 14, 15].

Decision questions for folding (unit) cubes are studied by Aichholzer et al. [2, 3]. The *half-grid model* allows folds of all degrees along the grid lines, the diagonals, as well as along the horizontal and vertical halving lines of the squares. In this model, every polyomino of size at least 10 folds into the cube [3]. The remaining polyominoes of smaller size are explored by Czajkowski et al. [8]. In the grid model, Aichholzer et al. [3] characterized the foldable tree-shaped polyominoes that fit within a $3 \times n$ strip. Investigating polyominoes with holes, Aichholzer et al. [2] show that all but five *basic* holes (a single unit square, a slit of length 1, a straight slit of length 2, a corner slit of length 2 and a U-shaped slit of length 3) guarantee that the polyomino folds in the grid model into the cube.

In the context of polyiamonds, Aichholzer et al. [3] present a nice and simple characterization of polyiamonds that fold into the smallest platonic solid: Even when restricting to folds along the grid lines, a polyiamond folds into the tetrahedron if and only if it contains one of the two tetrahedral nets.

Results and organization. Our main results are as follows: In Section 2, we identify some sufficient and necessary conditions for foldability and take a closer look at polyiamonds with slits and holes. Among our findings, we characterize foldable polyiamonds containing a hole of positive area: each but one polyiamond is foldable. In Section 3, we characterize the convex foldable polyiamonds: A convex polyiamond folds into $\mathcal{O}$ if and only if it contains one of five polyiamonds. Lastly, in Section 4, we show that every polyiamond of size ≥ 15 is foldable. An unfoldable polyiamond of size 14 proves that this bound is best possible.

2 First Thoughts

2.1 Foldability

A polyiamond P *contains* a polyiamond P' if P' can be translated, rotated, and reflected such that all triangles and triangle sides of P' also belong to P. Restricting our attention to the triangles, a polyiamond P $\triangle$-*contains* a polyiamond P' if all triangles of P' belong to P. As we will show in Observation 2.5, neither containment nor $\triangle$-containment of a foldable polyiamond is a sufficient folding criterion. Nevertheless, we are able to show two sufficient criteria based on $\triangle$-containment of foldable polyiamonds. By zig-zag-folding as indicated in Figure 2, every polyiamond can be reduced to a contained convex polyiamond.

▶ **Lemma 2.1.** *A polyiamond P is foldable if it $\triangle$-contains a convex foldable polyiamond C.*

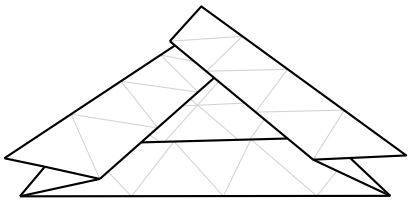


Figure 2 Folding strategy to reduce to convex subpolyiamonds by zig-zag-folds of the outside.

A *net* of a polyhedron is formed by cutting along certain edges and unfolding the resulting connected set to lie flat. There exist two interesting facts for nets of 3-dimensional regular convex polyhedra [7]: Firstly, each net is uniquely determined by a spanning tree of the 1-skeleton of the polyhedron, i.e., the cut edges form a spanning tree of the vertex-edge graph. Secondly, dual polyhedra (e.g., the cube and the octahedron) have the same number of nets. Consequently, there exist eleven octahedron nets. They are depicted in Figure 3.

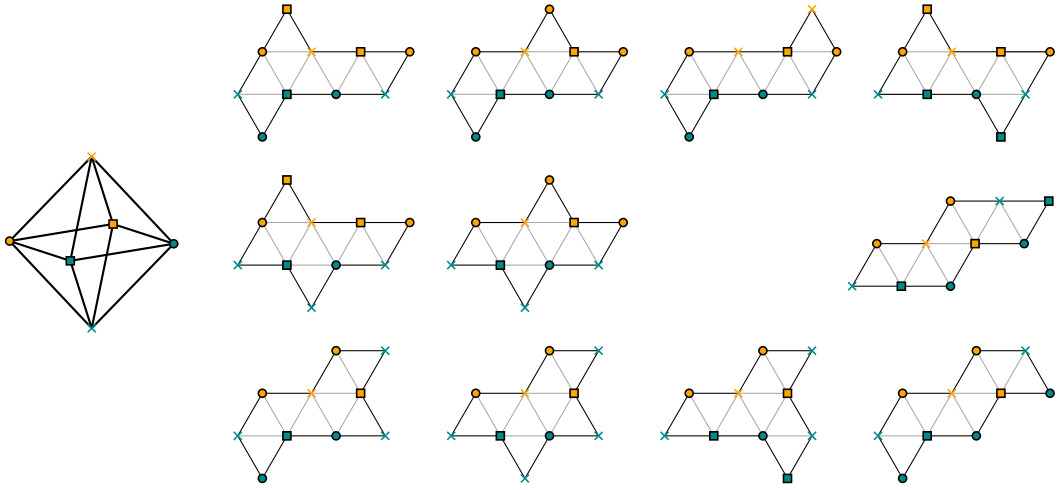


Figure 3 The eleven nets of the octahedron. In a folding, vertices of the same label are mapped to the same corner of $\mathcal{O}$.

We can show that $\triangle$-containing a net is a sufficient folding criterion for a polyiamond.

▶ **Lemma 2.2.** *A polyiamond is foldable if it $\triangle$-contains an octahedron net.*

2.2 Unfoldability

As indicated in Figure 4, the triangular grid graph allows for a proper 3-coloring. Because every (connected) inner triangulation has at most one 3-coloring (up to exchange of the colors), every polyiamond has a unique 3-coloring which is induced by the triangular grid. If there exists a slit edge along a grid line, the polyiamond graph may have several vertices corresponding to one grid vertex, see also Figure 6(b). Note that each corner of $\mathcal{O}$ has a unique non-adjacent corner which we call its *antipodal*.

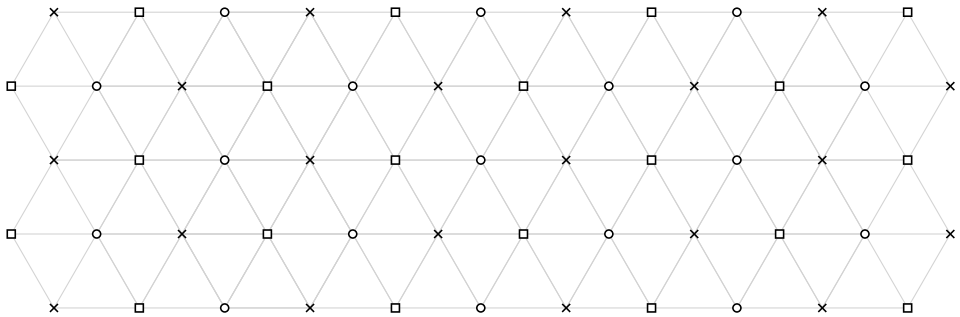


Figure 4 A 3-coloring of the triangular grid.

In order to study the unfoldability, we also consider partial foldings. In particular, when relaxing the condition that all faces are covered, we say a polyiamond is folded into a partial octahedron.

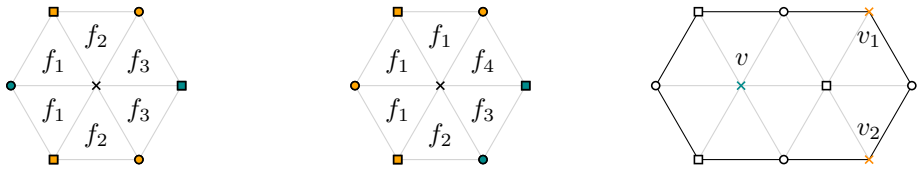
▶ **Lemma 2.3.** *Let P be a polyiamond with a 3-coloring of its vertices. In every folding of P to a (partial) octahedron, the vertices of each color class are mapped to (one vertex or a pair of) antipodal corners of $\mathcal{O}$.*

The proof of Lemma 2.3 follows from the following observation: Consider two neighboring triangles of P and note that their two private vertices have the same color. If their common side is folded by $\pm\beta$, the two vertices are mapped to antipodal corners of $\mathcal{O}$; otherwise the edge is folded by $\pm 180°$ and the two vertices are mapped to the same corner of $\mathcal{O}$.

Let C_6 and C_{10} denote the polyiamonds depicted in Figures 5(a) and 5(c), respectively. The following lemma is a crucial tool to disprove foldability.

▶ **Lemma 2.4.** *Let P be a polyiamond folded into a partial octahedron $\mathcal{O}$.*

- ***a.*** *Every C_6 $\triangle$-contained in P covers at most 4 different faces of $\mathcal{O}$.*
- ***b.*** *If a C_6 in P covers exactly 3 or 4 faces, then the induced triangle-face-mapping is unique (up to symmetry) and as depicted in Figures 5(a) and 5(b), respectively.*
- ***c.*** *Every C_{10} contained in P covers at most 6 different faces of $\mathcal{O}$.*



(a) A triangle-face-map of C_6 covering 3 faces of $\mathcal{O}$.

(b) A triangle-face-map of C_6 covering 4 faces of $\mathcal{O}$.

(c) Any triangle-face-map of C_{10} covers at most 6 faces of $\mathcal{O}$.

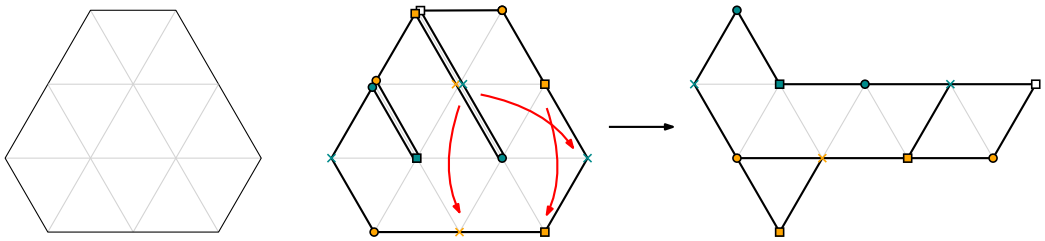
Figure 5 Illustration of Lemma 2.4 and its proof.

2.3 On Slits and Holes

Note that removing individual edges from a foldable polyiamond cannot destroy its foldability (if connectivity is maintained). This allows us to focus on polyiamonds without slit edges. We would like to remark that slits may in fact enable foldability.

▶ **Observation 2.5.** *Let P be a polyiamond ($\triangle$-)containing a foldable polyiamond P'. Then, the polyiamond P may not be foldable.*

As we show in Theorem 3.1, the polyiamond P depicted in Figure 6(a) does not fold into $\mathcal{O}$, while the polyiamond P' with additional slit edges in Figure 6(b) can be transformed into a polyiamond containing a net. Hence, P' is foldable by Lemma 2.2.



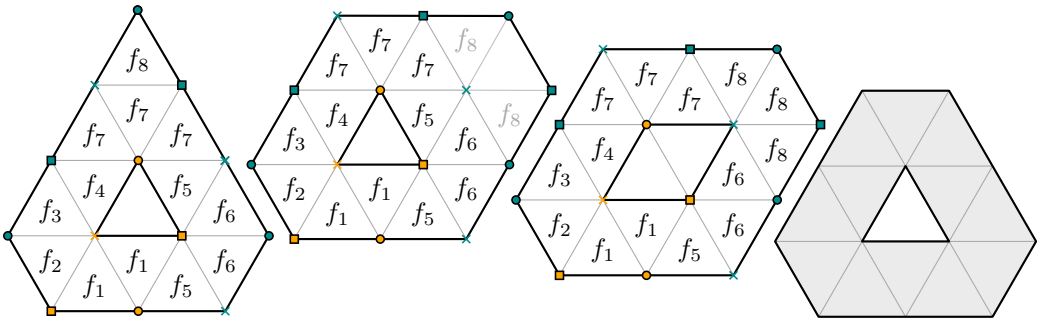
(a) This polyiamond does not fold into $\mathcal{O}$. (Theorem 3.1)

(b) With additional slit edges, the polyiamond folds into $\mathcal{O}$.

Figure 6 Illustration for Observation 2.5.

We now characterize foldable polyiamonds with holes of positive area. Let O denote the polyiamond illustrated in Figure 7(d).

▶ **Theorem 2.6.** *Let P be a polyiamond containing a hole h of positive area. Then P folds into $\mathcal{O}$ if and only if it is not the polyiamond O.*



(a) Polyiamond P_a **(b)** Polyiamond P_b **(c)** Polyiamond P_c **(d)** Polyiamond O

Figure 7 llustration for the proof of Theorem 2.6.

For the proof, we focus on a largest hole h of P and distinguish two cases: If h contains two neighboring triangles, P reduces (by zig-zag-folding) to the foldable polyiamond P_c depicted in Figure 7(c). Otherwise, we focus on an individual triangle contained in h and the case that P cannot be reduced to the foldable polyiamond P_a, see Figure 7(a). Then, zig-zag-folds yield a foldable subpolyiamond P' of P_b depicted in Figure 7(b). It follows that every polyiamond with a hole h of positive area and size ≥ 13 folds into $\mathcal{O}$.

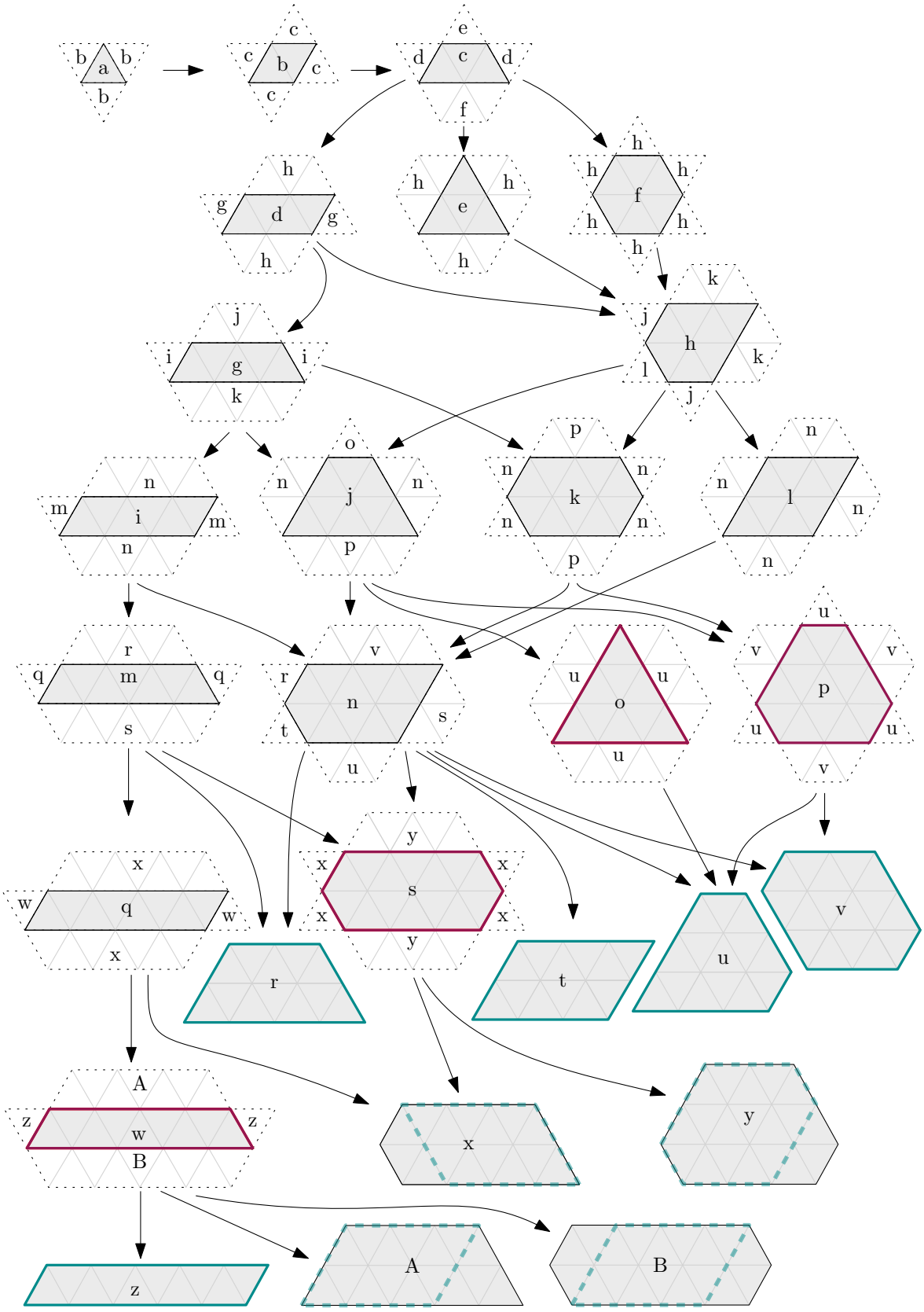


Figure 8 Construction of all $\mathcal{C}$-free polyiamonds; the inclusion-wise maximal $\mathcal{C}$-free polyiamonds o, p, s, and w are highlighted in red.

3 Characterization for Convex Polyiamonds

In this section, we characterize convex foldable polyiamonds. Let $\mathcal{C}$ denote the set of five convex polyiamonds depicted in Figure 9.

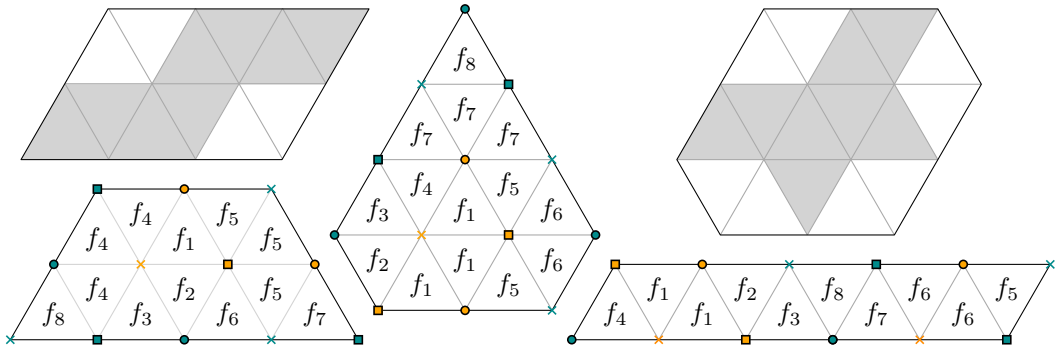


Figure 9 Illustration for Theorem 3.1; the set $\mathcal{C}$ of foldable polyiamonds and their foldings.

▶ **Theorem 3.1.** *A convex polyiamond P folds into $\mathcal{O}$ if and only if it contains one of the five polyiamonds in $\mathcal{C}$.*

In the proof we exploit the convexity and Lemma 2.1 for both directions. For one, it suffices to present folding strategies for the polyiamonds in $\mathcal{C}$, see Figure 9. For the other direction, we construct all convex $\mathcal{C}$-*free* polyiamonds (that contain no polyiamond in $\mathcal{C}$), see Figure 8, and show that the inclusion-wise maximal $\mathcal{C}$-free polyiamonds do not fold into $\mathcal{O}$.

4 A Sharp Size Bound

As shown in the proof of Theorem 3.1, the polyiamond s depicted in Figure 8 is not foldable, i.e., there exist polyiamonds of size 14 that do not fold into $\mathcal{O}$. In this section, we show the following complementing theorem.

▶ **Theorem 4.1.** *Every polyiamond P of size ≥ 15 folds into $\mathcal{O}$.*

To present an idea of the proof, we give some useful sufficient conditions and a simple upper bound. Let P be a polyiamond and ℓ some grid line. The ℓ-*width* of P denotes the size of the polyiamond obtained by folding all edges parallel to ℓ in a zig-zag-fashion as indicated in Figure 2. The *width* of P is the maximum of the three different ℓ-widths. Because the convex polyiamond $P_{-} := z$, depicted in Figure 8, folds into $\mathcal{O}$, we obtain the following.

▶ **Lemma 4.2.** *Every polyiamond P of width at least 10 folds into $\mathcal{O}$.*

Moreover, we determine an upper bound on the size of polyiamonds of width ≤ 9, see Figure 10 for the construction of the maximal polyiamonds of width at most 9. In particular, they have size ≤ 42 which yields a nice and simple upper bound.

▶ **Corollary 4.3.** *Every polyiamond of size > 42 folds into $\mathcal{O}$.*

To show the sharp bound, we need to work a little harder.

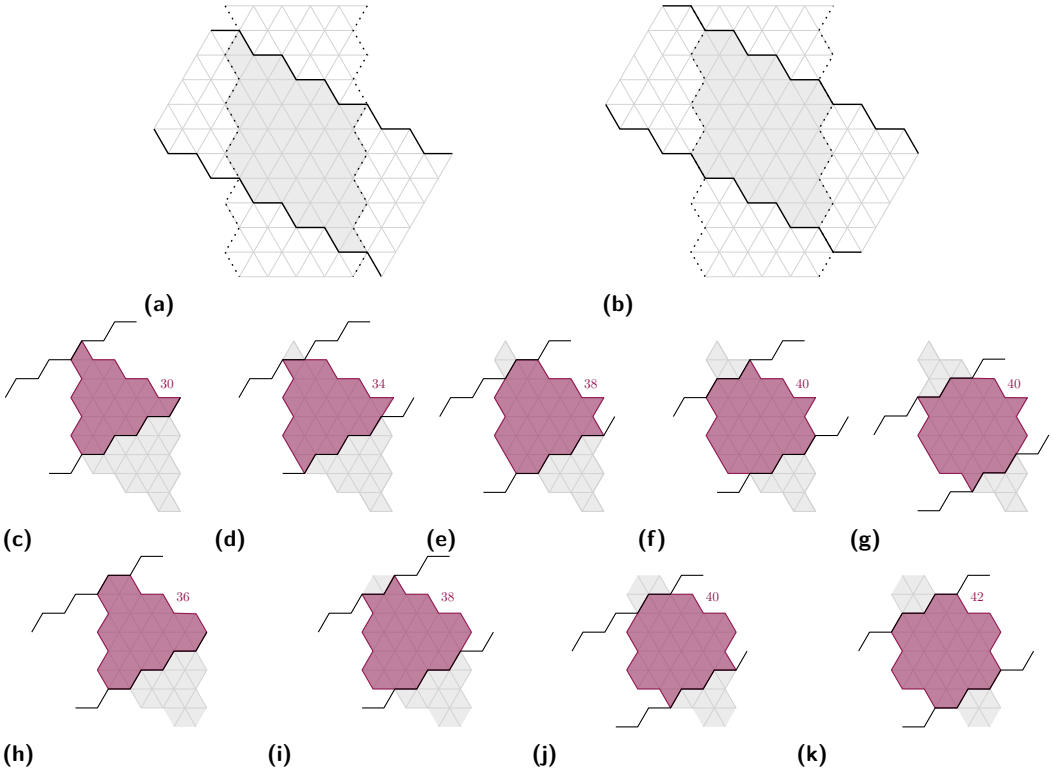
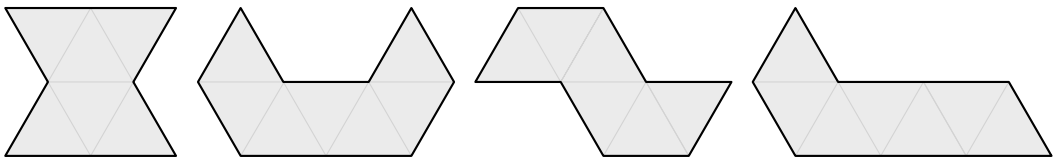


Figure 10 Illustration for the proof of Corollary 4.3. Construction of the maximal polyiamonds of width ≤ 9; their sizes are indicated by numbers.

Proof sketch for the sharp upper bound. Theorem 4.1 is based on a strong sufficient criterion. Let P_X, P_U, P_Z, and P_L denote the polyiamonds depicted in Figures 11(a) to 11(d), respectively.

▶ **Proposition 4.4.** *Every polyiamond P that $\triangle$-contains P_X, P_U, P_Z, or P_L and has size ≥ 15 folds into $\mathcal{O}$.*



(a) P_X **(b)** P_U **(c)** P_Z **(d)** P_L

Figure 11 Illustration of the four polyiamonds used in Proposition 4.4.

We call a polyiamond $\mathcal{P}$-*free* if it does not $\triangle$-contain any of the polyiamonds P_-, P_X, P_U, P_Z, or P_L. By Theorem 3.1 and Proposition 4.4, it remains to show that no $\mathcal{P}$-free polyiamond of size ≥ 15 exists. To do so, we construct all $\mathcal{P}$-free polyiamonds bottom-up with computer assistance and observe that indeed no such polyiamond exists.

Acknowledgments We thank Christian Rieck for valuable suggestions on a draft of this manuscript and the anonymous reviewers for constructive feedback. Additionally, the first author thanks Tilman Stehr for his good advice concerning questions of implementation.

References

1 Zachary Abel, Erik Demaine, Martin Demaine, Hiroaki Matsui, Günter Rote, and Ryuhei Uehara. Common developments of several different orthogonal boxes. In *Canadian Conference on Computational Geometry (CCCG '11)*, 2011.

2 Oswin Aichholzer, Hugo Akitaya, Kenneth Cheung, Erik Demaine, Martin Demaine, Sándor P. Fekete, Linda Kleist, Irina Kostitsyna, Maarten Löffler, Zuzana Masárová, and Klara Mundilova. Folding polyominoes with holes into a cube. *Computational Geometry (CGTA)*, 93:101700, 2020. doi:10.1016/j.comgeo.2020.101700.

3 Oswin Aichholzer, Michael Biro, Erik D. Demaine, Martin L. Demaine, David Eppstein, Sándor P. Fekete, Adam Hesterberg, Irina Kostitsyna, and Christiane Schmidt. Folding polyominoes into (poly)cubes. *International Journal of Computational Geometry & Applications (IJGCA)*, 28:197–226, 2018. doi:10.1142/S0218195918500048.

4 Greg Aloupis, Prosenjit K. Bose, Sébastien Collette, Erik D. Demaine, Martin L. Demaine, Karim Douïeb, Vida Dujmović, John Iacono, Stefan Langerman, and Pat Morin. Common unfoldings of polyominoes and polycubes. In *Computational Geometry, Graphs and Applications (CGGA '10)*, pages 44–54. Springer, 2010. doi:10.1007/978-3-642-24983-9_5.

5 Nadia M. Benbernou, Erik D. Demaine, Martin L. Demaine, and Anna Lubiw. Universal hinge patterns for folding strips efficiently into any grid polyhedron. *Computational Geometry*, page 101633, 2020. doi:10.1016/j.comgeo.2020.101633.

6 Nadia M. Benbernou, Erik D. Demaine, Martin L. Demaine, and Aviv Ovadya. Universal hinge patterns for folding orthogonal shapes. *Origami^5: Proceedings of the 5th International Conference on Origami in Science, Mathematics and Education (OSME '11)*, pages 405–419, 2011.

7 F. Buekenhout and M. Parker. The number of nets of the regular convex polytopes in dimension ≤ 4. *Discrete Mathematics*, 186(1):69 – 94, 1998. doi:10.1016/S0012-365X(97)00225-2.

8 K.Y. Czajkowski, Erik D. Demaine, Martin L. Demaine, K. Eppling, R. Kraft, Klara Mundilova, and L. Smith. Folding small polyominoes into a unit cube. In *Canadian Conference on Computational Geometry (CCCG '20)*, 2020.

9 E. Hawkes, B. An, N. M. Benbernou, H. Tanaka, S. Kim, E. D. Demaine, D. Rus, and R. J. Wood. Programmable matter by folding. *Proceedings of the National Academy of Sciences*, 107(28):12441–12445, 2010. doi:10.1073/pnas.0914069107.

10 Kaori Kuribayashi, Koichi Tsuchiya, Zhong You, Dacian Tomus, Minoru Umemoto, Takahiro Ito, and Masahiro Sasaki. Self-deployable origami stent grafts as a biomedical application of ni-rich tini shape memory alloy foil. *Materials Science and Engineering: A*, 419:131–137, 03 2006. doi:10.1016/j.msea.2005.12.016.

11 Jun Mitani and Ryuhei Uehara. Polygons folding to plural incongruent orthogonal boxes. In *Canadian Conference on Computational Geometry (CCCG '08)*, pages 39–42, 2008.

12 Ala Qattawi, Mahmoud Abdelhamid, Ahmad Mayyas, and Mohammed Omar. Design analysis for origami-based folded sheet metal parts. *SAE International Journal of Materials and Manufacturing*, 7(2):488–498, 2014. URL: http://www.jstor.org/stable/26268627.

13 Toshihiro Shirakawa and Ryuhei Uehara. Common developments of three incongruent orthogonal boxes. *International Journal of Computational Geometry & Applications (IJGCA)*, 23(01):65–71, 2013. doi:10.1142/S0218195913500040.

14 Ryuhei Uehara. A survey and recent results about common developments of two or more boxes. In *Origami^6: Proceedings of the 6th International Meeting on Origami in Science, Mathematics and Education (OSME '14)*, volume 1, pages 77–84, 2014.

15 Dawei Xu, Takashi Horiyama, Toshihiro Shirakawa, and Ryuhei Uehara. Common developments of three incongruent boxes of area 30. *Computational Geometry (CGTA)*, 64:1–12, 2017. doi:10.1016/j.comgeo.2017.03.001.

Computing Optimal Virtual Camera Trajectories

Kerem Geva[1], Matthew J. Katz[2], and Eli Packer[3]

1 **Ben-Gurion University of the Negev, Israel**
gevak@post.bgu.ac.il

2 **Ben-Gurion University of the Negev, Israel**
matya@cs.bgu.ac.il

3 **Intel Corporation, Israel**
eli.packer@intel.com

Abstract

We introduce algorithms for generating virtual camera trajectories for highly dynamic scenes. The input to our algorithms is the locations of objects forming the scene in each time frame of a given sequence of such frames. In each frame, we determine the location of a virtual camera, so that (i) the objects of interest are well captured, and (ii) the induced trajectory of the virtual camera has plausible geometric properties. The trajectories we create should facilitate the generation of good quality virtual clips that both show the main events of the underlying scene and are appealing to view.

1 Introduction

Immersive videos (also known as 360-degrees videos) are videos in which a view in every direction is recorded simultaneously by using a collection of cameras. By capturing scenes from various angles and applying Multi View Stereo techniques [3], one can create virtual videos that mimic immersive videos. This means that there are six degrees of freedom to determine the camera position, viewing direction and zoom-in level in each frame. Having this technology at hand, videos of virtual cameras have become popular in recent years. The virtual cameras trajectories are usually determined by administrators who run dedicated software. The resulting trajectories are usually very simple, e.g., straight lines or trajectories that follow a specific dynamic object.

Creating virtual videos is a very challenging task, mainly because it greatly depends on the 3D reconstruction of the objects in the scene. The latter is a non-precise task with many challenging difficulties. Just to name a few, it depends on precise camera calibrations, visibility of several cameras and the multi-view stereo algorithm. As those contain inherent errors, the results are never completely precise and visible errors are evident.

Another challenging task is to define good virtual camera trajectories. The challenge here mainly relates to finding trajectories that are plausible for the viewers, reveal the major interests of the scene and hide potential 3D reconstruction errors. Moreover, the geometry of the trajectory is usually crucial for producing good results. In many cases, it should be smooth, of uniform speed and without fast turns, etc.

In this work, we propose an algorithmic framework to create good camera trajectories. Our main goal is to generate trajectories such that the most interesting objects of the scene are visible, while certain plausible geometric features (such as those mentioned above) are maintained. We apply computational geometry tools and multi-objective optimization techniques to achieve this goal. As far as we know, we are the first to do so in this context.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

2 Overview

In this section we give an overview of our algorithm for the production of a virtual clip.

A clip is a sequence of k frames, where each frame captures a given simple scene. For each frame, we need to determine the location of the virtual camera from which to capture the given scene. This location should, on the one hand, enable a good view of the scene, and, on the other hand, it should fit well in the sequence of locations forming the clip.

Our algorithm consists of the following three stages. In the first stage we consider each frame separately. Given the scene for some frame f_i, we compute a set of simple suitable regions R_i around the scene. A region is considered suitable if any point in it is a suitable location for the virtual camera, in the sense that it enables a good view of the scene (assuming the camera is directed towards the center of the scene).

In the second stage we construct a layer graph G with k layers, one per frame. The nodes of the i'th layer are the regions of R_i, and the subgraph induced by any two consecutive layers is a complete weighted bipartite graph. The weight of an edge between $r \in R_i$ and $r' \in R_{i+1}$ is determined by several parameters, see details below. Finally, we add nodes s and t, and connect s to all the nodes of the first layer by edges of weight zero and connect t to all the nodes of the last layer by edges of weight zero.

By computing a minimum-weight path from s to t in G, we obtain a crude plan for our virtual camera. That is, if the computed path is $\pi = s, r_1, r_2, \ldots, r_k, t$, then the i'th frame in the clip will correspond to the view from some point p_i in the region r_i, for $i = 1, \ldots, k$.

The input to the third stage is thus the sequence of regions $(r_1, \ldots, r_k)$, and the goal of this stage is to select a location for the camera in each of these regions. The final clip will then consist of the sequence of views corresponding to these locations. We present two alternative approaches for selecting the locations, where the emphasis in both approaches is on the integration of the individual views.

In both approaches, we assume that the regions are simple polygons with some additional properties, and we need to traverse the polygons in the given order, such that the resulting path is (nearly) optimal in some sense. In the first approach, we are only interested in minimizing the total length of the path, which is (a special case of) the *touring polygons problem* [2], and in the second approach we are interested in optimizing several measures, including path length, number and sharpness of turns and uniform edge length. Thus, we term this approach as the *multi-objective touring polygons problem*. In this version of the paper, we only discuss the second approach.

3 The layer graph

In this section we describe the construction of the layer graph for a sequence of k frames. We focus on two main issues: determining the nodes of each layer of the graph and setting the edge weights.

Below we describe one of several options for determining the nodes of a layer. The choice of a specific option depends on the type of clip that we wish to create. Let c_i denote the "center" of the scene for the i'th frame (e.g. the position of the ball), and let O_i be the annulus centered at c_i with inner radius r and outer radius R. We restrict the location of the virtual camera to the annulus O_i. The radii r and R are fixed; they are determined by distance constraints of the system. On the one hand, the virtual camera should not be too close to the scene's center, so that inaccuracies in the reconstruction are not noticed by the viewer, and, on the other hand, the camera should not be too far from the scene, so that it is seen in sufficient detail, see Figure 1(a).

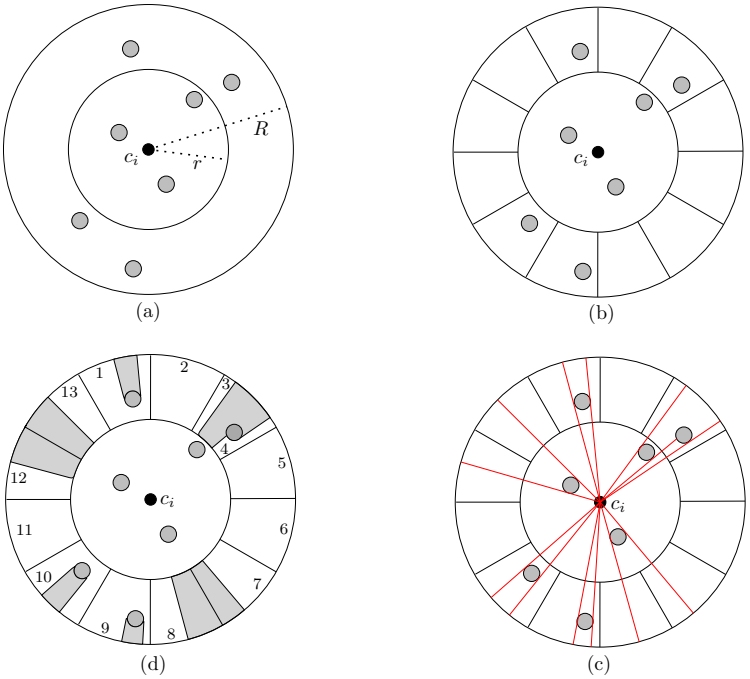


Figure 1 Constructing the set of suitable regions R_i.

Our goal is to define the set R_i of suitable regions within O_i. We begin by dividing O_i into m sectors, where in each sector the direction from any point in it to c_i is more or less fixed, but, the difference between the directions corresponding to two points from non-adjacent sectors is significant and noticeable; see Figure 1(b). The second step is to find the regions in O_i that should be avoided, in the sense that if the camera is placed in one of them, it will not capture the scene (and in particular c_i) well due to obstructions by other objects, e.g. some player; see Figure 1(c). After performing these two steps, we get a partition of (the good portion of) O_i into disjoint regions that can be further partitioned and refined into simple polygons. These polygons consist of the set of suitable regions R_i around the i'th scene; see Figure 1(d).

As stated in the overview, the set of nodes of the i'th layer, V_i, corresponds to the regions R_i, and $V = (\cup_{i=1}^{k} V_i) \cup \{s, t\}$, where V is the node set of the graph. Set $E_i = \{(u, v) | u \in V_i, v \in V_{i+1}\}$, for $1 \leq i \leq k-1$, and $E_0 = \{(s, v) | v \in V_1\}$ and $E_k = \{(v, t) | v \in V_k\}$. Then, the edge set of the graph is $E = E_0 \cup E_1 \cup \cdots \cup E_{k-1} \cup E_k$.

Let v_i^x be the vertex in layer i that corresponds to the region $r_x \in R_i$, and let $S(v_i^x)$ be the sector to which r_x belongs in O_i ($1 \leq S(v_i^x) \leq m$). We determine the weight of an edge $e \in E$ as follows:

- $w(e) = 0$ for each $e \in E_0 \cup E_k$.
- $w(e) = 0$ for each $e = (v_i^x, v_{i+1}^y)$ such that $r_x \cap r_y \neq \emptyset$
- Otherwise, the weight of $e = (v_i^x, v_{i+1}^y)$ is determined by several parameters including the distance between r_x and r_y, and the difference between $S(v_i^x)$ and $S(v_{i+1}^y)$. So that, the edges connecting regions in different sectors will have higher weight than those connecting regions in the same sector, as well as the edges connecting distant regions will have higher weight than those connecting nearby regions.

We compute a minimum-weight path from s to t in $G = (E, V, W)$, and get a path

$p = (s, v_1^{x_1}, v_2^{x_2}, \ldots, v_k^{x_k}, t)$ that visits in order exactly one vertex in each layer of the graph. The i'th frame in our clip will correspond to the view from some point p_i in the region r_{x_i} for $i = 1, \ldots, k$.

4 Multi-objective shortest polygon tours

Recall that once a suitable region in each frame is chosen (see Section 2), we wish to find a trajectory with some desirable properties that visits these regions in order. Next, we list the features that we wish to optimize.

- **Length**. Each trajectory vertex corresponds to the position of the camera in a specific frame, and, in general, shorter trajectories allow slower movement of the camera and thus a more plausible viewing experience.
- **Number of links**. Each trajectory vertex introduces a sharp turn. We could smooth the trajectories, but this will not remove the zigzagging effect which is less plausible to view. Hence, minimizing the number of links may help in decreasing this effect. Finding touring polygons of minimum links was studied in [4].
- **Uniform speed**. As frames come at constant rate, minimizing the differences in edge length will result in more uniform speed. The latter is important for smoothing the virtual clips.
- **Angle turns**. Minimizing the angles between consecutive trajectory edges helps diminish the turns of the camera and thus improve the overall quality.

In order to optimize the above criteria, we apply the multi-objective shortest path problem (MOSP) algorithm by Breugem et al. [1] (see also, [5]). The input to their algorithm is a graph, where each edge is associated with a vector of d non-negative integer weights. Each of the d weights of an edge corresponds to a different objective one would like to minimize. The output of the algorithm is a set of 'good' paths, where each path is characterized by the sum vector, which is the sum of the vectors of its edges, i.e., the sum of the weights on the path's edges for each of the d objectives. They find a subset of all possible paths which they term the *Pareto-optimal frontier*. Those are all the paths for which it is impossible to improve one objective without harming the others and thus they constitute the locally optimal paths. In general, the best path depends on the priorities of the various objectives. For any path π and objective z, let π_z be the total weights of objective z on the edges of π. As the size of the Pareto-optimal frontier is exponential in the worst case (the problem is in fact NP-complete), Breugem et al. propose a FPTAS with approximation factor ϵ_2, for any given $\epsilon_2 > 0$. Their idea is to find a collection of paths Q that approximates all Pareto-optimal paths in the following sense. For any Pareto-optimal path π there is a path $\pi' \in Q$, such that for any objective z, $\pi'_z \leq (1 + \epsilon_2)\pi_z$. They show that the size of Q is polynomial in the input.

In order to use MOSP, we transform our problem to a shortest path problem on a grid. In each suitable region we lay a grid of size $\epsilon_1 > 0$. The grid points in the suitable regions consist of the set of vertices V of the graph G. The edges of G correspond to the Cartesian products of the vertex sets of consecutive regions (with an exception that we mention below). We formulate the above objectives (except for the first which is obvious) as a shortest path problem as follows.

- **Number of links**. The appropriate cost on each edge will be 1 to reflect the use of one link. We also add edges between non-consecutive regions of weight 1 if they pass through all intermediate ones.
- **Uniform speed**. The appropriate cost on each edge will be the square of its length to discourage the use of long edges.

- **Angle turns**. As this criterion has strong correlation with the shortest distance objective, we ignore it in our formulation.

Once MOSP produces the almost-Pareto-optimal paths (up to ϵ_2), we can choose the one that serves the predefined priorities on the objectives.

The running time of MOSP is $O(nm(n\log(nC)/\epsilon_2)^{d-1})$, where n is the number of vertices, m is the number of edges, C is the square of the length of the longest edge after normalizing the distances, and d is the number of objectives. Suppose that our input consists of k frames and the area of each region is at most w, then in our setting $n = O(kw/\epsilon_1^2)$ and $m = O((kw/\epsilon_1^2)^2)$ (or $m = O(k(w/\epsilon_1^2)^2)$ if we do not have edges between non-consecutive regions), and the overall running time is $T(k, w, \epsilon_1, \epsilon_2) = O((kw/\epsilon_1^2)^3((kw/\epsilon_1^2)\log(kwC/\epsilon_1^2)/\epsilon_2)^{d-1})$.

5 Future work

We are planning to implement the framework described in this paper and to run the program that is obtained on real-world data from sport events. The idea is to render the clips from the computed virtual camera trajectories and to compare them with clips obtained with manual or semi-manual techniques. We plan to perform a plethora of experiments with various objectives and parameters.

Moreover, in Section 3 we described one (somewhat simplistic) way to determine the nodes of a layer of the layer graph. We intend to generalize it by, e.g., allowing some control over the camera's orientation, so that it is not necessarily determined by its location with respect to the center c_i.

Acknowledgments

The authors would like to thank J.S.B Mitchell and Alon Efrat for helpful discussions.

References

1 Thomas Breugem, Twan Dollevoet, and Wilco van den Heuvel. Analysis of FPTASes for the multi-objective shortest path problem. *Comput. Oper. Res.*, 78:44–58, 2017. URL: https://doi.org/10.1016/j.cor.2016.06.022, doi:10.1016/j.cor.2016.06.022.

2 Moshe Dror, Alon Efrat, Anna Lubiw, and Joseph S. B. Mitchell. Touring a sequence of polygons. In Lawrence L. Larmore and Michel X. Goemans, editors, *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, June 9-11, 2003, San Diego, CA, USA*, pages 473–482. ACM, 2003. URL: https://doi.org/10.1145/780542.780612, doi:10.1145/780542.780612.

3 Yasutaka Furukawa and Carlos Hernández. Multi-view stereo: A tutorial. *Found. Trends Comput. Graph. Vis.*, 9(1-2):1–148, 2015. URL: https://doi.org/10.1561/0600000052, doi:10.1561/0600000052.

4 Leonidas J. Guibas, John Hershberger, Joseph S. B. Mitchell, and Jack Snoeyink. Approximating polygons and subdivisions with minimum link paths. *Int. J. Comput. Geom. Appl.*, 3(4):383–415, 1993. URL: https://doi.org/10.1142/S0218195993000257, doi:10.1142/S0218195993000257.

5 George Tsaggouris and Christos D. Zaroliagis. Multiobjective optimization: Improved FPTAS for shortest paths and non-linear objectives with applications. *Theory Comput. Syst.*, 45(1):162–186, 2009. URL: https://doi.org/10.1007/s00224-007-9096-4, doi:10.1007/s00224-007-9096-4.

Polygon-Universal Graphs

Tim Ophelders[1], Ignaz Rutter[2], Bettina Speckmann[1], and Kevin Verbeek[1]

1 **Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands**
{t.a.e.ophelders|b.speckmann|k.a.b.verbeek}@tue.nl

2 **Department of Computer Science and Mathematics, University of Passau, Germany**
rutter@fim.uni-passau.de

Related Version A full version of the paper is available at arxiv.org/abs/2103.06916.

1 Introduction

We study a fundamental question from graph drawing: given a pair (G, C) of a graph G and a cycle C in G together with a simple polygon P, is there a straight-line drawing of G inside P which maps C to P? We say that such a drawing of (G, C) *respects* P (see Fig. 1). We fully characterize those instances (G, C) which are *polygon-universal*, that is, they have a drawing that respects P for any simple (not necessarily convex) polygon P. Specifically, we identify two necessary conditions for an instance to be polygon-universal. Both conditions are based purely on graph and cycle distances and are easy to check (see Section 2). In the remainder of the paper we show that these two conditions are also sufficient. If an instance (G, C) is planar, that is, if there exists a planar drawing of G with C on the outer face, we show that the same conditions guarantee for every simple polygon P the existence of a planar drawing of (G, C) that respects P. If (G, C) is polygon-universal, then our proofs directly imply a linear-time algorithm to construct a drawing that respects a given polygon P.

Related work Tutte [5] proved that there is a straight-line planar drawing of a planar graph G inside an arbitrary convex polygon P if one fixes the outer face of (an arbitrary planar embedding of) G to P. This result has been generalized to allow polygons P that are non-strictly convex [1, 2] or even star-shaped polygons [3]. These results have applications in *partial drawing extension* problems. Here, in addition to an input graph G, we are given a subgraph $H \subseteq G$ together with a fixed drawing Γ of H. The question is whether one can extend the given drawing Γ to a planar straight-line drawing of the whole graph G by

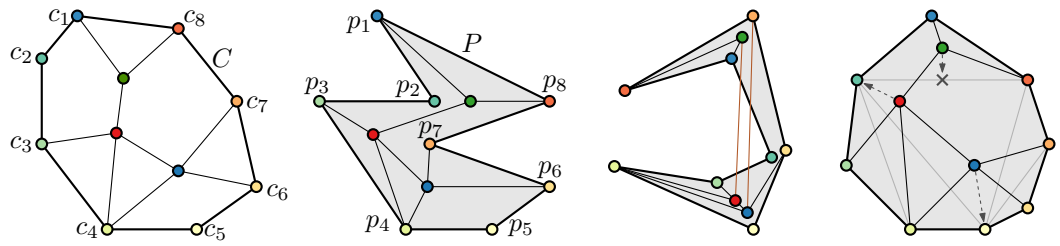


Figure 1 Left: an instance (G, C). Center left: a drawing of (G, C) that respects a polygon P (shaded in grey). Center right: there is no drawing of (G, C) that respects this polygon. Right: A triangulated convex polygon with a drawing that is not triangulation-respecting; moving the vertices along the dashed arrows results in a triangulation-respecting drawing.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

drawing the vertices and edges of $G - H$ inside the faces of H. If the embedding of G is fixed, the results by Tutte and others allow to reduce the problem by removing vertices of G that are contained in convex or star-shaped faces of Γ. Such reduction rules have led to efficient testing algorithms for special cases, for example, when the drawing of H is convex [4].

Notation Let $G = (V, E)$ be a graph with n vertices. A *drawing* $\mathcal{D}$ of G is a map from each $v \in V$ to points in the plane and from each edge $e \in E$ to a Jordan arc connecting its endpoints. A *straight-line* drawing maps each edge to a straight line segment. A drawing is *planar*, if no two edges intersect, except at common endpoints. A graph G is planar if it has a planar drawing. Let $C = [c_1, \ldots, c_t]$ with $c_i \in V$ be a simple cycle in G. An *instance* (G, C) is planar if G has a planar drawing with C as the outer face. Let P be a simple polygon with t vertices $[p_1, \ldots, p_t]$ with $p_i \in \mathbb{R}^2$. A drawing $\mathcal{D}$ of (G, C) *respects* P if it is a map $\mathcal{D}\colon V \to P$ from vertices to points in P such that $\mathcal{D}(c_i) = p_i$ and for each edge $\{u, v\} \in E$, the line segment between $\mathcal{D}(u)$ and $\mathcal{D}(v)$ lies in P (see Figure 1). That is, $\mathcal{D}$ is a straight-line drawing of G inside P that fixes the vertices of C to the corresponding vertices of P. An instance (G, C) is *(planar) polygon-universal* if it admits a (planar) straight-line drawing that respects every simple (not necessarily convex) polygon P on t vertices.

Our algorithms use a triangulation $\mathcal{T}$ of P to construct a drawing or prove the non-universality. We say that a drawing of (G, C) *respects* $\mathcal{T}$ if no edge of G properly crosses an edge of $\mathcal{T}$. Although every triangulation-respecting drawing is also polygon-respecting, the converse is not true. In fact, there are graphs G, cycles C, and polygons P that have a polygon-respecting drawing, but no triangulation of P admits a triangulation-respecting drawing (see Figure 2).

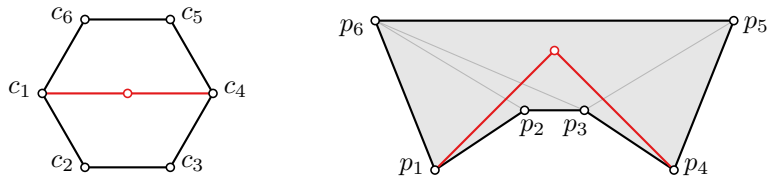


Figure 2 A graph G and a polygon for which there exists a drawing of G, but no triangulation with a triangulation-respecting drawing.

2 Necessary conditions for polygon-universality

We present two necessary conditions for an instance (G, C) to be polygon-universal. Intuitively, both conditions capture the fact that there need to be "enough" vertices in G between cycle vertices for the drawing not to become "too tight". The *Pair Condition* captures this fact for any two vertices on the cycle C. The *Triple Condition* is a bit more involved: even if the Pair Condition is satisfied for any pair of vertices on the cycle, there can still be triples of vertices which together "pull too much" on the graph. Specifically, for an instance (G, C) of a graph G and a cycle $C \subseteq G$ with t vertices, we denote by $d_G\colon V \times V \to \mathbb{N}_0$ the graph distance in G and by $d_C\colon V(C) \times V(C) \to \mathbb{N}_0$ the distance (number of edges) along the cycle C. The following conditions are necessary for (G, C) to be polygon-universal for all simple polygons P:

Pair For all i and j, we have $d_C(c_i, c_j) \leq d_G(c_i, c_j)$ (and hence $d_C(c_i, c_j) = d_G(c_i, c_j)$).

Triple For all vertices $v \in V$ and distinct i, j, k with $d_C(c_i, c_j) + d_C(c_j, c_k) + d_C(c_i, c_k) \geq t$ (and hence $= t$), we have $d_G(c_i, v) + d_G(c_j, v) + d_G(c_k, v) > t/2$.

To establish that these two conditions are necessary, we use the *link distance* between two points inside certain simple polygons P. Specifically, the link distance of two points q_1 and q_2 with respect to a simple polygon P is the minimum number of segments for a polyline π that lies inside P and connects q_1 and q_2. If the Pair Condition is violated for two cycle vertices c_i and c_j, we can construct a *Pair Spiral* polygon P (see Figure 3 (left)) such that the link distance between p_i and p_j (the vertices of P to which c_i and c_j are mapped) exceeds $d_G(c_i, c_j)$. Clearly there is no drawing (G, C) that respects P.

If the first condition holds, but the second condition is violated by a vertex v and three vertices c_i, c_j, c_k of the cycle C, then the distance along C between any pair of c_i, c_j, c_k is the same as the shortest path between them that passes through v. That is, $d_C(c_i, c_j) = d_G(c_i, v) + d_G(c_j, v)$, $d_C(c_j, c_k) = d_G(c_j, v) + d_G(c_k, v)$ and $d_C(c_i, c_k) = d_G(c_i, v) + d_G(c_k, v)$. In that case, we can construct a *Triple Spiral* polygon P (see Figure 3 (right)) such that there is no point that lies within link-distance $d_G(c_i, v)$ from c_i, link distance $d_G(c_j, v)$ from c_j, and link distance $d_G(c_k, v)$ from c_k simultaneously. Hence, there exists no drawing of the aforementioned shortest paths through v that respects P.

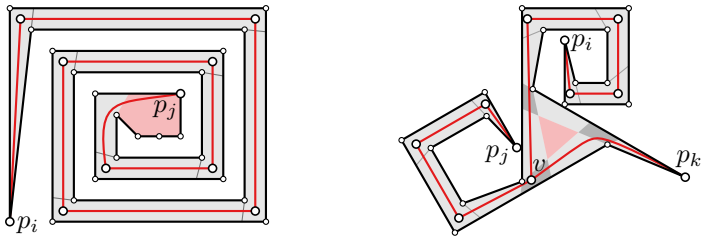


Figure 3 Left: Pair Spiral. Points with link-distance greater than $d_G(c_i, c_j)$ from p_i shaded red. Right: Triple Spiral. Points of link-distance $\leq d_G(c_x, v)$ from p_x for one $x \in \{i, j, k\}$ in light gray; for two $x \in \{i, j, k\}$ in dark gray; there is no point q in P with $d_G(c_x, q) \leq d_G(c_x, v)$ for all $x \in \{i, j, k\}$.

3 Triangulation-respecting drawings

In this section we are given the following input: an instance (G, C) consisting of a graph G with n vertices and a cycle C with t vertices, and a simple polygon P with t vertices together with an arbitrary triangulation $\mathcal{T}$ of P. We study the following question: is there a drawing of (G, C) that respects both P and $\mathcal{T}$?

We describe a dynamic programming algorithm which can answer this question in linear time. The basic idea is as follows: every edge of $\mathcal{T}$ defines a *pocket* of P. We recursively *sketch* a drawing of G within each pocket. Such a sketch assigns an approximate location, such as an edge or a triangle, to each vertex. Ultimately we combine the location constraints on vertex positions posed by the sketches and decide if they can be satisfied.

We root (the dual tree of) $\mathcal{T}$ at an arbitrary triangle T_{root}. Each edge e of $\mathcal{T}$ partitions P into two regions, one of which contains T_{root}. Let Q be the region not containing T_{root}. We say that Q is a *pocket* with the *lid* $e = e_Q$, and we denote the unique triangle outside Q adjacent to e_Q by T_Q^+. Since a pocket is uniquely defined by its lid, we will for an edge e also write Q_e to denote the pocket with lid e. We say that a pocket is *trivial* if its lid lies on the boundary of P; in such case the pocket consists of only that edge. If Q is a non-trivial pocket, then we denote the unique triangle inside Q adjacent to e_Q by T_Q (see Figure 4).

We first define triangulation-respecting drawings for pockets: a triangulation-respecting drawing for a pocket Q with lid $e_Q = (p_i, p_j)$ is an assignment of the vertices of G to locations inside the polygon P, such that (i) any vertex c_ℓ with $i \leq \ell \leq j$ is assigned to the polygon

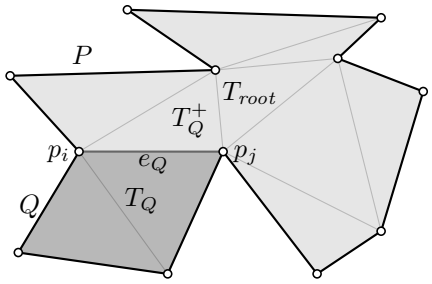


Figure 4 A triangulation with labels for the pocket Q (shaded dark) and triangles T_Q and T_Q^+ incident to edge e_Q of the triangulation.

vertex p_ℓ and (ii) for any edge f of G, the vertices of f lie on a common triangle (or edges or vertices thereof). Note that we consider the triangles of the triangulation as closed, and hence two distinct triangles may share a segment (namely an edge of the triangulation) or a point (namely a vertex of P). We define a triangulation-respecting drawing for the entire triangulation analogously, requiring that c_ℓ is assigned to p_ℓ for all ℓ.

A *sketch* is an assignment of the vertices of G to simplices (vertices, edges, or triangles) of the triangulation with the property that, if we draw each vertex anywhere on its assigned simplex, then the result is a triangulation-respecting drawing. We hence interpret a simplex as a closed region of the plane. Formally, a sketch of the triangulation is a function Γ that assigns vertices of G to simplices of $\mathcal{T}$, such that (i) for any vertex c_i of the cycle, $\Gamma(c_i) = p_i$, and (ii) for any two adjacent vertices u and v, there exists a triangle of $\mathcal{T}$ that contains both $\Gamma(u)$ and $\Gamma(v)$. A sketch of a pocket is defined similarly, except that vertices p_i of the polygon that lie outside the pocket do not need c_i assigned to them. We show that a sketch exists (for a pocket or a triangulation) if and only if there is a triangulation-respecting drawing (for that pocket or triangulation). If a pocket admits a sketch, we call the pocket *sketchable*. If a particular pocket is sketchable, then so are all of its subpockets, because any sketch for a pocket is automatically a sketch for any of its subpockets.

We present an algorithm that for any sketchable pocket constructs a sketch, and for any other pocket reports that it is not sketchable. This algorithm recursively constructs particularly well-behaved sketches for child pockets, and combines these sketches into a new well-behaved sketch. To obtain a sketch for $\mathcal{T}$, we combine the three well-behaved sketches for the three pockets whose lids are the edges of the root triangle T_{root} – assuming that all three pockets are sketchable.

▶ **Theorem 1.** *There is a linear-time algorithm to decide if (G, C) has a triangulation-respecting drawing for a simple polygon P with fixed triangulation $\mathcal{T}$; the same algorithm also constructs a drawing if one exists.*

4 Planar triangulation-respecting drawings

We are given the same input as in Section 3, namely an instance (G, C) consisting of a graph G with n vertices and a cycle C with t vertices, and a simple polygon P with t vertices together with an arbitrary triangulation $\mathcal{T}$ of P. In addition, we assume that the instance (G, C) is planar, that is, G has a planar drawing $\mathcal{D}$ with C on the outer face. Note that $\mathcal{D}$ does not necessarily map vertices of C to vertices of P.

Analogously to Section 3, we can ask the following question: is there a planar drawing of (G, C) that respects both P and $\mathcal{T}$? However, the answer to this question is often 'no',

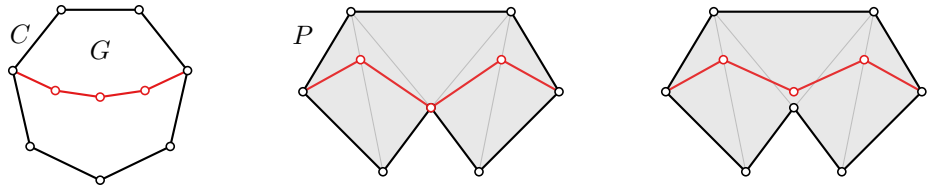


Figure 5 Left: A planar instance. Center: a triangulation-respecting drawing in which two vertices coincide. Right: a perturbed drawing that is planar but not triangulation-respecting.

even when both triangulation-respecting drawings and planar polygon-respecting drawings exist. Consider, for example, Figure 5: a planar triangulation-respecting drawing for this combination of (G, C), P, and $\mathcal{T}$ does not exist; any drawing inside P either places two vertices on top of each other, or edges cross edges of the triangulation. Nonetheless, triangulation-respecting drawings are a useful tool for our final goal of constructing planar polygon-respecting drawings. For example, the triangulation-respecting drawing of Figure 5 can be perturbed infinitesimally to obtain a planar polygon-respecting drawing (that is not triangulation-respecting). We show that if a planar instance (G, C) has a triangulation-respecting drawing, then it also has a *weakly-planar triangulation-respecting* one, that is, a triangulation-respecting drawing that is planar and polygon-respecting after infinitesimal perturbation. (That is, vertices may be moved to a simplex of $\mathcal{T}$ that contains the original location.) Hence, the algorithm described in Section 3 can decide for a planar instance (G, C) whether there is a weakly-planar triangulation-respecting drawing.

Consider now a planar drawing $\mathcal{D}$ of (G, C). We call the triple $(G, C, \mathcal{D})$ a *plane instance*. In the following, we create weakly-planar triangulation-respecting drawings $\mathcal{W}$, such that a planar polygon-respecting perturbation $\widetilde{\mathcal{W}}$ of $\mathcal{W}$ "mimics" $\mathcal{D}$ inside P. More specifically, $\widetilde{\mathcal{W}}$ is isotopic to $\mathcal{D}$ in the plane, that is, one can be continuously deformed into the other without introducing crossings. We say that $\mathcal{W}$ *accommodates* $(\mathcal{D}, \mathcal{T})$. A plane instance $(G, C, \mathcal{D})$ is *sketchable* if (G, C) has a sketch (for $\mathcal{T}$). Recall here, that a sketch does not have a notion of planarity. However, we show in Theorem 2 that any sketchable plane instance $(G, C, \mathcal{D})$ has a drawing $\mathcal{W}$ which accommodates $(\mathcal{D}, \mathcal{T})$.

▶ **Theorem 2.** *A plane instance $(G, C, \mathcal{D})$ has a drawing that accommodates $(\mathcal{D}, \mathcal{T})$ if and only if $(G, C, \mathcal{D})$ is sketchable.*

The algorithm implied by Theorem 1 can check in linear time if a plane instance $(G, C, \mathcal{D})$ has a sketch and via Theorem 2 the same algorithm can decide in linear time if $(G, C, \mathcal{D})$ has an accommodating drawing. This drawing can be constructed in polynomial time.

5 Sufficient conditions for polygon-universality

In Section 2 we proved that the Pair and Triple Conditions are necessary for an instance (G, C) to be polygon-universal. In Sections 3 and 4 we argued that an instance (G, C) has a triangulation-respecting drawing for a triangulation $\mathcal{T}$ of P if and only if it has a sketch for $\mathcal{T}$. We can show that the Pair Condition alone already implies that each pocket has a sketch. The Triple Condition then allows us to combine sketches at the root T_{root} of $\mathcal{T}$.

▶ **Theorem 3.** *Let (G, C) be an instance that satisfies the Pair and Triple Conditions. Then (G, C) has a triangulation-respecting drawing for any triangulation of any simple polygon.*

▶ **Corollary 4.** *Let $(G, C, \mathcal{D})$ be a plane instance that satisfies the Pair and Triple Conditions. Then (G, C) has a drawing that accommodates $(\mathcal{D}, \mathcal{T})$ for any triangulation $\mathcal{T}$ of any simple polygon.*

6 Discussion and Conclusion

We have characterized the (planar) polygon-universal graphs (G, C) by means of simple combinatorial conditions involving (graph-theoretic) distances along the cycle C and in the graph G. In particular, this shows that, even though the recognition of polygon-universal graphs most naturally lies in $\forall\exists\mathbb{R}$, it can in fact be tested in polynomial time, by explicitly checking the Pair and the Triple Conditions. Our main open question concerns the restriction to simple polygons without holes. Can a similar characterization be achieved in the presence of holes? Or is the polygon-universality problem for simple polygons with holes $\forall\exists\mathbb{R}$-complete?

Another interesting question concerns the running time for recognizing polygon-universal graphs. Testing the Pair and Triple Conditions naively requires at least $\Omega(n^3)$ time. On the other hand, at least in the non-planar case, given (G, C) and a polygon P with arbitrary triangulation T, we can in linear time either find an extension or a violation of the Pair/Triple Condition, which shows that the instance is not polygon-universal. (Recall that a polygon-extension for P might exist, though not one that respects T, see Figure 2). For planar instances, the contraction to minimal instances causes an additional linear factor in the running time. Can (planar) polygon-universality be tested in $o(n^2)$ time?

References

1 Erin W. Chambers, David Eppstein, Michael T. Goodrich, and Maarten Löffler. Drawing graphs in the plane with a prescribed outer face and polynomial area. *Journal of Graph Algorithms and Applications*, 16(2):243–259, 2012. doi:10.7155/jgaa.00257.

2 Christian A. Duncan, Michael T. Goodrich, and Stephen G. Kobourov. Planar drawings of higher-genus graphs. *Journal of Graph Algorithms and Applications*, 15(1):7–32, 2011. doi:10.1007/978-3-642-11805-0_7.

3 Seok-Hee Hong and Hiroshi Nagamochi. Convex drawings of graphs with non-convex boundary constraints. *Discrete Applied Mathematics*, 156(12):2368–2380, 2008. doi:10.1016/j.dam.2007.10.012.

4 Tamara Mchedlidze, Martin Nöllenburg, and Ignaz Rutter. Extending convex partial drawings of graphs. *Algorithmica*, 76(1):47–67, 2016. doi:10.1007/s00453-015-0018-6.

5 William T. Tutte. How to draw a graph. *Proceedings of the London Mathematical Society*, 13(3):743–768, 1963. doi:10.1112/plms/s3-13.1.743.

Rectilinear Steiner Trees in Narrow Strips

Henk Alkema[1] and Mark de Berg[2]

1 **Department of Mathematics and Computer Science, TU Eindhoven, the Netherlands**
h.y.alkema@tue.nl

2 **Department of Mathematics and Computer Science, TU Eindhoven, the Netherlands**
m.t.d.berg@tue.nl

Abstract

A *rectilinear Steiner tree* for a set P of points in $\mathbb{R}^2$ is a tree that connects the points in P using horizontal and vertical line segments. The goal of Minimum Rectilinear Steiner Tree is to find a rectilinear Steiner tree with minimal total length. We investigate how the complexity of Minimum Rectilinear Steiner Tree for point sets P inside the strip $(-\infty, +\infty) \times [0, \delta]$ depends on the strip width δ. We give an algorithm with running time $n^{O(\sqrt{\delta})}$ for sparse point sets, where each $1 \times \delta$ rectangle inside the strip contains $O(1)$ points.

1 Introduction

In the Minimum Steiner Tree problem in the plane, we are given as input a set P of points in the plane, called *terminals*, and the goal is to find a minimum-length tree that connects the terminals in P. Thus the given terminals must be nodes of the tree, but the tree may also use so-called *Steiner points* as nodes. Minimum Steiner Tree is a classic optimization problem. It was among the first problems to be proven NP-hard, not only for the case where the length of the tree is measured using Euclidean metric [10] but also in the rectilinear version [11]. It was also shown to be NP-hard for other metrics [5]

The rectilinear version of the problem, where the edges of the tree must be horizontal or vertical, is one of the most widely studied variants, and it is also the topic of our paper. The Minimum Rectilinear Steiner Tree problem dates back more than 50 years [12, 13]. Its popularity arises from its many applications, in particular in the design of integrated circuits [6, 3, 4, 19]. The two most important early insights on Minimum Rectilinear Steiner Tree came from Hanan [13] and Hwang [15]. Hanan observed that any terminal set P admits a minimum rectilinear Steiner tree (MRST, for short) whose edges lie on the grid formed by all horizontal and vertical lines passing through at least one terminal in P. This grid is often called the *Hanan grid*. This implies that the Minimum Rectilinear Steiner Tree problem can be reduced to a purely combinatorial problem—namely, a Steiner-tree problem on graphs—which is not possible for the Euclidean version of the problem. Hwang provided a characterization of the different components of an MRST.

As mentioned, Minimum Rectilinear Steiner Tree can be considered a special case of the Steiner-tree problem on graphs. Here the input is an edge-weighted graph $G = (V(G), E(G))$ and a terminal set $P \subseteq V(G)$, and the goal is to compute a minimum-length subtree of G that includes all terminals. In 1971 Dreyfus and Wagner [8] gave an algorithm solving the Steiner-tree problem on graphs in time $3^n \cdot \log W \cdot |V(G)|^{O(1)}$, where W is the maximum edge weight in G. This was improved by Björklund *et al.* [2] and Nederlof [17], to $2^n \cdot W \cdot |V(G)|^{O(1)}$. A variant of the Dreyfus-Wagner algorithm for Minimum Rectilinear Steiner Tree runs in time $O(n^2 \cdot 3^n)$. Thobmorson *et al.* [18] and

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Deneen *et al.* [7] gave randomized algorithms for the special case of Minimum Rectilinear Steiner Tree where the terminals are drawn independently and uniformly from a rectangle. Both algorithms run in $2^{O(\sqrt{n}\log n)}$ expected time. Finally, Fomin *et al.* [9] presented a $2^{O(\sqrt{n}\log n)}$ algorithm for general (non-random) point sets in 2018.

Due to the many applications of Minimum Steiner Tree variants in the plane, there has also been significant interest in practical implementations. These implementations rely on the insight that a minimum Steiner tree can always be decomposed into so-called *full components*, which are maximal subtrees that do not have any terminals as internal nodes [14]. (This holds for the Euclidean as well as the rectilinear version.) To compute an exact solution, a set of candidate full components is first computed and then it is computed which subset of candidate full components can be concatenated into an MRST. This process was introduced by Winter in 1985 [20], in his software package *GeoSteiner*. Throughout the years, GeoSteiner, which has become a collaboration between Warme, Winter and Zachariasen, has remained the fastest publicly available software package for computing minimum Steiner trees in the plane. By 2018, it could solve instances for up to 4,000 points for the rectilinear version, and up to 10,000 points for the Euclidean version [16].

Our contribution. The fastest known algorithm for Minimum Rectilinear Steiner Tree in $\mathbb{R}^2$ runs in $2^{O(\sqrt{n}\log n)}$ time [9]. In $\mathbb{R}^1$, on the other hand, the problem can be solved in $O(n\log n)$ time by just sorting the points. To better understand the computational complexity of the classic Minimum Rectilinear Steiner Tree problem in the plane, we investigate how the complexity depends on the width of the terminal set P. If P is "almost 1-dimensional" in the sense that the points lie in a narrow strip $\mathbb{R}\times[0,\delta]$, then can we solve Minimum Rectilinear Steiner Tree more efficiently than in the general case? If so, how does the complexity scale with δ? Can we obtain an algorithm that is fixed-parameter tractable with respect to δ? This follows the line of research started recently by Alkema *et al.* [1], who studied these questions for the Traveling Salesman Problem. To be able to use that the points lie in a narrow strip, we need a further assumption, namely that for any $x\in\mathbb{R}$ the rectangle $[x,x+1]\times[0,\delta]$ contains $O(1)$ points. We show that for these *sparse point sets* in $\mathbb{R}^2$ there exists an MRST that intersects any vertical line only $O(\sqrt{\delta})$ times. We also give a dynamic-programming algorithm that runs in $n^{O(\sqrt{\delta})}$ time. In the full version of this paper, we also give an algorithm with $\min\{n^{O(\sqrt{\delta})}, 2^{O(\delta\sqrt{\delta})}n\}$ expected running time for points generated randomly inside a rectangle of height δ and expected width n.

2 Preliminaries

Let $P := \{p_1,\ldots,p_n\}$ be a set of *terminals* in a 2-dimensional strip with height δ —we call such a strip a *δ-strip*—which we assume without loss of generality to be $\mathbb{R}\times[0,\delta]$. We use x_i and y_i to denote the x- and y-coordinate of point p_i, respectively. The points can be sorted on their x-coordinates in $O(n\log n)$ time. Therefore, we will from now on assume that $x_i \leqslant x_j$ for all $1\leqslant i\leqslant j\leqslant n$. We denote the vertical distance between two horizontal edges e, e' (or the horizontal distance between two vertical edges) by $\mathrm{dist}(e,e')$.

Next we give some (mostly standard) terminology concerning rectilinear Steiner trees. A *rectilinear tree* is a tree structure embedded in the plane whose edges are horizontal or vertical line segments overlapping only at their endpoints. The *length* of a tree T, or $\|T\|$, is the sum of the lengths of its edges. A *rectilinear Steiner tree* for a set P of terminals is a rectilinear tree such that each terminal $p\in P$ is an endpoint of an edge in the tree. A *minimal rectilinear Steiner minimal tree (MRST)* is such a tree of minimum length.

Separators will play a crucial role in our algorithm. A *separator* is a vertical line, not

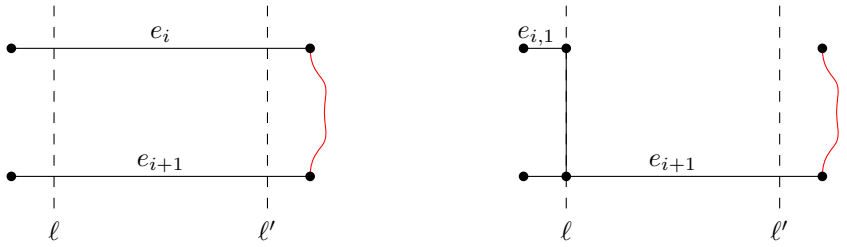


Figure 1 Illustration for the proof of Observation 2. On the left, T. On the right, T'. Since $\text{dist}(e_i, e_{i+1}) < \text{dist}(\ell, \ell')$, the tree T' is shorter than T.

containing any of the points in P, that separates P into two non-empty subsets. For all $1 \leqslant i < n$ such that $x_i < x_{i+1}$, we define s_i to be the separator with x-coordinate $(x_i + x_{i+1})/2$. The *tonicity* of a rectilinear tree T *at a separator* s is the number of times T crosses s; The *tonicity* of a rectilinear tree T is the maximum over the tonicity of T at all separators. Finally, a well-known property of the MRST is the following:

▶ Observation 1. [Hanan [13]] Let P be a set of terminals in $\mathbb{R}^2$. Then there exists an MRST on P that is a subset of the *Hanan grid*, the grid formed by taking all horizontal and vertical lines which pass through at least one of the points of P.

From now on, we will only consider rectilinear Steiner trees that lie on the Hanan grid. Furthermore, we can now directly conclude that the tonicity of an MRST is at most n.

3 Sparse point sets inside a narrow strip

We say a point set is *sparse* if for all x the rectangle $[x, x+1] \times [0, \delta]$ contains at most k points for some arbitrary but fixed *sparseness constant* k. In this section, we will give a $n^{O(\sqrt{\delta})}$ algorithm for sparse point sets. We will do so in two steps. First, we will show that all separators are crossed at most $O(\sqrt{\delta})$ times. Then, we will give a dynamic-programming algorithm running in the desired time.

We will start by showing that parallel edges of an MRST cannot be too close.

▶ Observation 2. Let $E = \{e_1, \ldots, e_m\}$ be a set of m horizontal edges of an MRST T which all intersect two vertical lines ℓ and ℓ'. Then $m \leqslant 1 + \lfloor \delta / \text{dist}(\ell, \ell') \rfloor$. A similar statement holds when E is a set of vertical edges intersecting two horizontal lines.

Proof. W.l.o.g., let the edges in E be numbered from top to bottom, and let ℓ lie to the left of ℓ'. Suppose for a contradiction that $m > 1 + \lfloor \delta / \text{dist}(\ell, \ell') \rfloor$. Since $\text{dist}(e_1, e_m) \leqslant \delta$, there are two edges e_i and e_{i+1} such that $\text{dist}(e_i, e_{i+1}) \leqslant \delta/(m-1) < \text{dist}(\ell, \ell')$. We will now create a rectilinear Steiner tree T' strictly shorter than T, giving the desired contradiction. To this end we first delete the part of e_i between ℓ and ℓ'. Let $e_{i,1}$ denote the part of e_i to the left of ℓ (if any) and let $e_{i,2}$ denote the part of e_i to the right of ℓ' (if any); see Figure 1. The deletion splits T into two components. Assume without loss of generality that $e_{i,2}$ is in the same component as e_{i+1}. By deleting $e_{i,2}$ and connecting $e_{i,1}$ to e_{i+1} with a vertical edge contained in ℓ, we create a rectilinear Steiner Tree T' such that

$$\|T'\| = \|T\| - \text{dist}(\ell, \ell') - |e_{i,2}| + \text{dist}(e_i, e_{i+1}) < \|T\|,$$

giving the desired contradiction. ◀

When combined with the characterization of Hwang [15], this leads to the following lemma.

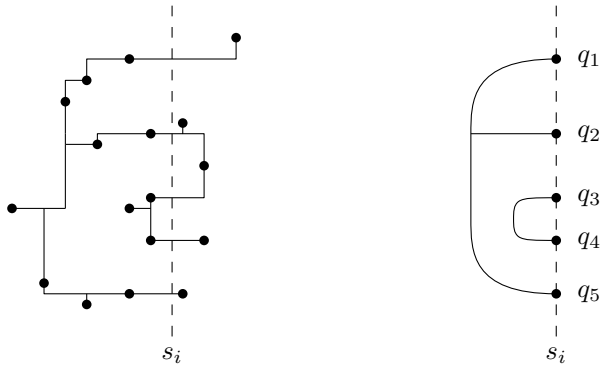


Figure 2 An example of an MRST and its crossing pattern $C = \{\{q_1, q_2, q_5\}, \{q_3, q_4\}\}$ at s_i

▶ **Lemma 3.** *Let P be a sparse point set in a δ-strip. Then there exists a $\left((9k+18)(2+\sqrt{\delta})\right)$-tonic MRST on P, where k is the sparseness constant.*

Proof. As the full proof is rather long, it can be found in the full version. We give a proof sketch below. Let s be a separator. Hwang previously showed that there exists an MRST such that each so-called full component has one of four different shapes [15]. For each of these shapes, every point can only be 'responsible' for a constant number of edges crossing s. Since the point set is sparse, the more edges cross s, the longer the average edge crossing s therefore has to be. We can then use Observation 2 to show that T is $O(\sqrt{\delta})$-tonic at s. ◀

Lemma 3 gives rise to a dynamic-programming algorithm, as explained next. Let T be a rectilinear Steiner tree, and let s_i be a separator. We define the *crossing pattern* of T at s_i as follows. Let $X(s_i)$ be the set of at most n points where the Hanan grid crosses s_i, and let $X(s_i, T) \subseteq X(s_i)$ be the subset of points where T crosses s_i. If T is an MRST,

$$|X(s_i, T)| \leqslant (9k+18)(2+\sqrt{\delta}) = O(\sqrt{\delta})$$

by Lemma 3. We partition $X(s_i, T)$ into parts (that is, subsets) such that two points from $X(s_i, T)$ are in the same part if the path in T between these points fully lies to the left of s_i. The resulting partition of $X(s_i, T)$ is the crossing pattern of T at s_i; see Figure 2 for an example. We will say that a rectilinear forest T *adheres to* C at s_i if T lies fully to the left of s_i, and there exists a rectilinear forest T' which lies fully to the right of s_i such that $T \cup T'$ is a rectilinear Steiner tree with crossing pattern C at s_i. Note that not all crossing patterns can lead to an MRST: those that require crossing edges on the left-hand side (because they do not have a proper "nesting structure") can never lead to an MRST. We call the crossing patterns that contain at most $(9k+18)(2+\sqrt{\delta})$ points and do not require crossing edges on the left-hand side *viable crossing patterns*. We will now count the number of viable crossing patterns at s_i. There are $n^{O(\sqrt{\delta})}$ possible sets $X(s_i, T)$ that contain at most $(9k+18)(2+\sqrt{\delta})$ points. The number of viable partitions of these points—also known as the number of non-crossing partitions—follows the Catalan numbers. Hence, there are $2^{O(\sqrt{\delta})}$ possible viable partitions for each $X(s_i, T)$. This implies that the total number of viable crossing patterns for s_i is $n^{O(\sqrt{\delta})} \cdot 2^{O(\sqrt{\delta})} = n^{O(\sqrt{\delta})}$.

The algorithm. We can now define a table entry $A[i, X]$ for each separator s_i and viable crossing pattern X at s_i as follows.

$$A[i, X] := \text{the minimum length of a rectilinear forest adhering to } X \text{ at } s_i.$$

Note that the length of an MRST equals $A[n, \{\emptyset\}]$. Next we describe a recursive formula to compute the table entries. As a base case, we will use $A[0, X] = 0$ for $X = \{\emptyset\}$, and ∞ for all other X.

Let s_j and s_i be consecutive separators, with $j < i$. Note that since the point set is sparse, at most k points share an x-coordinate. Therefore, $j \geqslant i - k$. Let $F(X, s_i)$ be a minimum-length rectilinear forest adhering to X at s_i, and let X' be its (unknown) crossing pattern at s_j. Then the value of $A[i, X]$ equals the value of $A[j, X']$ plus the total length of the edges of $F(X, s_i)$ between s_j and s_i. The total length of $F(X, s_i)$ between these two separators only depends on X' and X. Since this subproblem contains $O(\sqrt{\delta})$ points with three different x-coordinates, its Hanan grid contains only $O(\sqrt{\delta})$ edges. Therefore, its value can be computed in $2^{O(\sqrt{\delta})}$ time by simply checking every possible subset of edges. Let $L(X', X)$ denote the total length of the solution to this subproblem. If no solution exists, we define it to be ∞. Then we get

$$A[i, X] = \min_{\text{viable } X'} A[j, X'] + L(X', X),$$

where s_j is the separator immediately preceding s_i, and the sum is over all crossing patterns X' that are viable at s_j.

The running time. We first determine the number of table entries. There are $O(n)$ separators, and we have already seen that for every s_i there are $n^{O(\sqrt{\delta})}$ possible viable crossing patterns. Hence, the total number of table entries is $n^{O(\sqrt{\delta})}$. Next, we calculate the time needed per table entry. For each of the $n^{O(\sqrt{\delta})}$ possible viable crossing patterns X' we compute $L(X', X)$ in $2^{O(\sqrt{\delta})}$ time. This brings the total time needed per table entry to $n^{O(\sqrt{\delta})}$.

Since we have $n^{O(\sqrt{\delta})}$ table entries, each needing $n^{O(\sqrt{\delta})}$ time, we conclude:

▶ **Theorem 4.** *Let P be a sparse point set of size n inside a δ-strip. Then we can compute an MRST on P in $n^{O(\sqrt{\delta})}$ time.*

4 Concluding remarks

We proved that for sparse point sets in a strip of width δ, an MRST can be found in $n^{O(\sqrt{\delta})}$ time. We wonder whether an algorithm with running time $2^{O(\sqrt{\delta}\log\delta)} \cdot \text{poly}(n)$ is possible. For $\delta = \Theta(n)$ the running time would then equal the $2^{O(\sqrt{n}\log n)}$ of the algorithm for arbitrary point sets in the plane [9]. Another direction for future research is to study the problem in higher dimensions. We believe that our algorithmic results may carry over to $\mathbb{R}^d$ to points that are almost collinear, that is, that lie in a narrow cylinder. Generalizing the results to, say, points lying in a narrow slab will most likely be more challenging.

References

1 Henk Alkema, Mark de Berg, and Sándor Kisfaludi-Bak. Euclidean TSP in narrow strips. In *Proc. 36th International Symposium on Computational Geometry (SoCG 2020)*, volume 164 of *LIPIcs*, pages 4:1–4:16, 2020. doi:10.4230/LIPIcs.SoCG.2020.4.

2 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Fourier meets Möbius: fast subset convolution. In *Proc. 39th Annual ACM Symposium on Theory of Computing (STOC 2007)*, pages 67–74. ACM, 2007. doi:10.1145/1250790.1250801.

3 Marcus Brazil, Doreen A. Thomas, Jia F. Weng, and Martin Zachariasen. Canonical forms and algorithms for Steiner trees in uniform orientation metrics. *Algorithmica*, 44(4):281–300, 2006. doi:10.1007/s00453-005-1178-6.

4 Marcus Brazil and Martin Zachariasen. Steiner trees for fixed orientation metrics. *J. Glob. Optim.*, 43(1):141–169, 2009. doi:10.1007/s10898-008-9305-y.

5 Marcus Brazil and Martin Zachariasen. The uniform orientation Steiner tree problem is NP-hard. *Int. J. Comput. Geom. Appl.*, 24(2):87–106, 2014. doi:10.1142/S0218195914500046.

6 Marcus Brazil and Martin Zachariasen. *Optimal Interconnection Trees in the Plane*, volume 29. Springer, 05 2015. doi:10.1007/978-3-319-13915-9.

7 Linda Deneen, Gary Shute, and Clark Thomborson. A probably fast, provably optimal algorithm for rectilinear Steiner trees. *Random Structures & Algorithms*, 5:535 – 557, 10 1994. doi:10.1002/rsa.3240050405.

8 S. E. Dreyfus and R. A. Wagner. The steiner problem in graphs. *Networks*, 1(3):195–207, 1971. doi:10.1002/net.3230010302.

9 Fedor Fomin, Daniel Lokshtanov, Sudeshna Kolay, Fahad Panolan, and Saket Saurabh. Subexponential algorithms for rectilinear Steiner tree and arborescence problems. *ACM Transactions on Algorithms*, 16:1–37, 03 2020. doi:10.1145/3381420.

10 M. R. Garey, R. L. Graham, and D. S. Johnson. The complexity of computing Steiner minimal trees. *SIAM Journal on Applied Mathematics*, 32(4):835–859, 1977. URL: http://www.jstor.org/stable/2100193.

11 M. R. Garey and D. S. Johnson. The rectilinear Steiner tree problem is NP-complete. *SIAM Journal on Applied Mathematics*, 32(4):826–834, 1977. URL: http://www.jstor.org/stable/2100192.

12 E. N. Gilbert and H. O. Pollak. Steiner minimal trees. *SIAM Journal on Applied Mathematics*, 16(1):1–29, 1968. URL: http://www.jstor.org/stable/2099400.

13 M. Hanan. On Steiner's problem with rectilinear distance. *SIAM Journal on Applied Mathematics*, 14(2):255–265, 1966. URL: http://www.jstor.org/stable/2946265.

14 F. K. Hwang. On Steiner minimal trees with rectilinear distance. *SIAM Journal on Applied Mathematics*, 30(1):104–114, 1976. URL: http://www.jstor.org/stable/2100587.

15 F. K. Hwang and Dana S. Richards. Steiner tree problems. *Networks*, 22(1):55–89, 1992. doi:10.1002/net.3230220105.

16 Daniel Juhl, David Warme, Pawel Winter, and Martin Zachariasen. The geoSteiner software package for computing Steiner trees in the plane: an updated computational study. *Mathematical Programming Computation*, 10:487–532, 2018. doi:10.1007/s12532-018-0135-8.

17 Jesper Nederlof. Fast polynomial-space algorithms using inclusion-exclusion. *Algorithmica*, 65(4):868–884, 2013. doi:10.1007/s00453-012-9630-x.

18 Clark D. Thomborson, Linda L. Deneen, and Gary M. Shute. Computing a rectilinear Steiner minimal tree in $n^{O(\sqrt{n})}$ time. In *Proc. International Workshop on Parallel Algorithms and Architectures*, volume 269 of *Lecture Notes in Computer Science*, pages 176–183, 1987. doi:10.1007/3-540-18099-0_44.

19 Peter Widmayer, Ying-Fung Wu, and C. K. Wong. On some distance problems in fixed orientations. *SIAM J. Comput.*, 16(4):728–746, 1987. doi:10.1137/0216049.

20 Pawel Winter. An algorithm for the steiner problem in the euclidean plane. *Networks*, 15(3):323–345, 1985. doi:10.1002/net.3230150305.

A density-based metric learning approach to geometric inference *

Eugenio Borghini[1], Ximena Fernández[2], Pablo Groisman[3], and Gabriel Mindlin[4]

1 **Departamento de Matemática and IMAS-CONICET, FCEN, Universidad de Buenos Aires, Argentina.** eborghini@dm.uba.ar
2 **Department of Mathematics, Swansea University, UK and Departamento de Matemática, FCEN, Universidad de Buenos Aires, Argentina.** x.l.fernandez@swansea.ac.uk
3 **Departamento de Matemática, IMAS-CONICET, FCEN, Universidad de Buenos Aires, Argentina and NYU-ECNU Institute of Mathematical Sciences at NYU Shanghai.** pgroisma@dm.uba.ar
4 **IFIBA, CONICET and Departamento de Física, FCEN, Universidad de Buenos Aires, Argentina.** gabo@df.uba.ar

Abstract

We address the problem of estimating intrinsic distances in a manifold from a finite sample. We prove that the metric space defined by the sample endowed with a computable metric known as sample Fermat distance converges almost surely in the sense of Gromov–Hausdorff. The limiting object is the manifold itself endowed with the population Fermat distance, an intrinsic metric that accounts for both the geometry of the manifold and the density that produces the sample. The benefits of using Fermat distance as compared with the traditional Euclidean distance are illustrated by concrete applications in manifold learning and persistent homology.

Related Version arXiv:2012.07621

1 Introduction

Let $\mathbb{X}_n$ be a set of n independent sample points with common density f supported on a smooth manifold $\mathcal{M}$ embedded in $\mathbb{R}^D$. We assume that both $\mathcal{M}$ and f are unknown. The goal of manifold learning is to recover information about f and $\mathcal{M}$ from $\mathbb{X}_n$. Given an intrinsic density-based metric ρ in $\mathcal{M}$ (the Fermat distance), we are interested in finding computable estimators ρ_n of ρ over the sample $\mathbb{X}_n$ such that the metric space $(\mathbb{X}_n, \rho_n)$ is a good estimator of $(\mathcal{M}, \rho)$ in the sense of Gromov–Hausdorff.

The problem of learning intrinsic distances from samples has a long history. It is a crucial step in several learning tasks, such as finding low dimensional representations of data embedded in a possibly high dimensional Euclidean space, clustering and topology learning. Persistent homology is a central computational technique in Topological Data Analysis [2, 8, 9, 15, 18] developed to infer topological features from a sample, encoding valuable information about the shape of the underlying space. The results generated by this algorithm depend strongly on the notion of distance associated to the data.

Under the manifold assumption, intrinsic distances based on the distribution that produced the data capture both the geometry of the underlying manifold and the density of the point cloud, which can be convenient in presence of noise and outliers.

* This work was supported by the EPSRC grant New Approaches to Data Science: Application Driven Topological Data Analysis EP/R018472/1.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

In this article, we present a density-based Riemannian metric called Fermat distance together with a computable estimator based on a finite sample. We study the consistency of the estimator, as well as its Gromov–Hausdorff convergence. Finally, we show applications of the use of this estimator to overcome some weaknesses of classical algorithms in manifold learning and persistent homology. A full version of this manuscript (including proofs) is available at [3].

2 Density-based distance learning

Let $\mathcal{M}$ be a smooth closed Riemannian manifold without boundary of dimension $d > 1$ with Riemannian metric tensor g together with a continuous positive C^∞ density function $f : \mathcal{M} \to \mathbb{R}_{>0}$. For $p > 1$, consider the deformed metric tensor $g_p = f^{2(1-p)/d} g$. Since f is smooth, g_p is a Riemannian metric tensor. Thus, $\mathcal{M}$ has a metric space structure given by the geodesic distance with respect to g_p, denoted by $d_{f,p}$. The distance $d_{f,p}$ is called *deformed distance under* g_p in [13] and also *population Fermat distance* in [11].

▶ **Definition 2.1.** [11, 13] For $p > 1$, the *population Fermat distance* between $x, y \in \mathcal{M}$ is

$$d_{f,p}(x,y) = \inf_{\gamma} \int_I \frac{1}{f(\gamma_t)^{(p-1)/d}} \sqrt{g(\dot\gamma_t, \dot\gamma_t)} dt \tag{1}$$

where the infimum is taken over all piecewise smooth curves $\gamma \colon I \to \mathcal{M}$ with $\gamma_0 = x$, $\gamma_1 = y$.

In the special case when f is uniform, the population Fermat distance reduces to (a multiple of) the inherited Riemannian distance $d_{\mathcal{M}}$ from the ambient Euclidean space. When this is not the case, this distance takes into account the density. Indeed, geodesics in $\mathcal{M}$ respect to the distance $d_{f,p}$ are more likely to lie in regions with high values of f. The name Fermat distance comes from the analogy with optics, in which $d_{f,p}$ is the optical distance as defined by Fermat principle when the refraction index is given by $f^{-(p-1)/d}$.

Consider now a set $\mathbb{X}_n = \{x_1, x_2, \ldots, x_n\} \subseteq \mathcal{M}$ of n independent sample points in $\mathcal{M}$ with common density f. Suppose that $\mathcal{M}$ is embedded in $\mathbb{R}^D$ and it is endowed with the standard inherited Riemannian metric. Our aim is to approximate $d_{f,p}(x,y)$, assuming no knowledge about $\mathcal{M}$ and the Riemannian distance defined on it. To achieve this, we will define an estimator for this distance over the sample. We denote by $|x-y|$ the Euclidean distance between points $x, y \in \mathcal{M}$.

▶ **Definition 2.2.** [11, 14] For $p > 1$, the *sample Fermat distance* between $x, y \in \mathcal{M}$ is defined as

$$d_{\mathbb{X}_n,p}(x,y) = \inf_{\gamma} \sum_{i=0}^{r} |x_{i+1} - x_i|^p$$

where the infimum is taken over all paths $\gamma = (x_0, x_1, \ldots, x_{r+1})$ of finite length with $x_0 = x$, $x_{r+1} = y$ and $\{x_1, x_2, \ldots, x_r\} \subseteq \mathbb{X}_n$.

▶ **Example 2.3** (Eyeglasses)**.** We illustrate the effect of the Fermat distance $d_{\mathbb{X}_n,p}$ in a manifold $\mathcal{M}$ embedded in $\mathbb{R}^D$ by considering the *eyeglasses* curve in $\mathbb{R}^2$, uniformly sampled and perturbed with Gaussian noise, Figure 1. We compute the sample Fermat distance between each pair of points for a series of values of $p > 1$ and embed the sampled points in $\mathbb{R}^2$ in such a way that the Euclidean distance in the embedding reflects the Fermat distance, using the Multidimensional Scaling algorithm (MDS)[17]. As p becomes larger, the geometry of the data overcomes the bottleneck region and it deforms into a circle. We also compute the Isomap [1, 16] embedding in $\mathbb{R}^2$ for different values of the parameter k, the number

of nearest neighbors used to compute the neighborhood graph. Due to the noise near the bottleneck region, some points that are far in the sense of the inherited Riemannian distance become close in the distance estimated by Isomap. We show only the case $k = 10$ but similar results are obtained for different values of k. Note that, even if both Isomap distance and the sample Fermat distance converge to an intrinsic distance, the first one is independent of the density while the second one favors high density regions. As a consequence, Isomap embedding is sensitive to noise, while with Fermat distance the points lying in low density regions are mapped to points that are far from the rest of the sample (see the red points in Figure 1). The larger the power p, the stronger this effect. This feature allows Fermat distance to recover the underlying geometry of the manifold, even with noise.

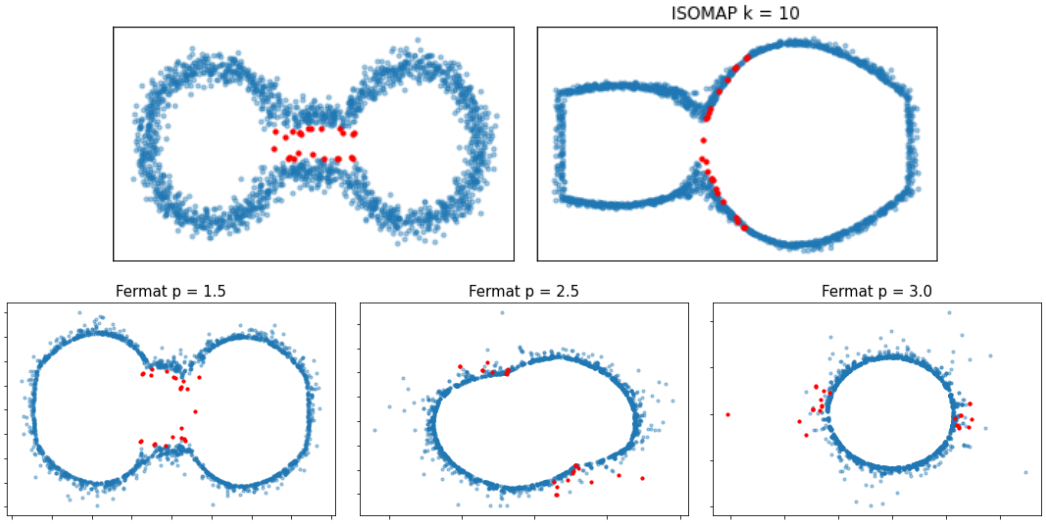


Figure 1 Top: A sample with noise of 2000 points of the eyeglasses dataset and Isomap projection with $k = 10$. Bottom: MDS embedding in $\mathbb{R}^2$ using Fermat distance for $p = 1.5, 2.5, 3.0$.

▶ Remark (Complexity.). The computation of the matrix of pairwise sample Fermat distances between points in $\mathbb{X}_n$ has complexity $\mathcal{O}(n^3)$ (but can be reduced to $\mathcal{O}(n^2 \log^2 n)$ with high probability) [11].

Our first result, Theorem 2.4, shows that the sample Fermat distance converges to the population Fermat distance for closed (i.e. compact and without boundary) submanifolds of $\mathbb{R}^D$. This was previously known for isometrically embedded (closures of) open sets of $\mathbb{R}^d$, [11]. Here we extend this result to a general class of manifolds and prove that the convergence is *uniform* (not only pointwise, as stated in [11]).

▶ **Theorem 2.4.** *For every $p > 1$ and $\lambda \in \big((p-1)/pd, 1/d\big)$, given $\varepsilon > 0$ there exist $\mu, \theta > 0$ such that, for n large enough*

$$\mathbb{P}\left(\sup_{x,y\in\mathcal{M}} \left| n^{(p-1)/d} d_{\mathbb{X}_n,p}(x,y) - \mu d_{f,p}(x,y) \right| > \varepsilon \right) \leq \exp\left(-\theta n^{(1-\lambda d)/(d+2p)}\right).$$

The constant μ depends only on p and d and is defined in [12].

Our goal is to compare globally the metric spaces $(\mathbb{X}_n, d_{\mathbb{X}_n,p})$ and $(\mathcal{M}, d_{f,p})$. In this sense, the relevant metric is the Gromov–Hausdorff distance, that allows to measure how far apart are two metric spaces from each other. We show that the sample $\mathbb{X}_n$ endowed

with a re-scaling of the sample Fermat distance converges to the metric space given by the manifold $\mathcal{M}$ with the (population) Fermat distance, in the sense of Gromov–Hausdorff.

▶ **Theorem 2.5.** *Let $\varepsilon > 0$ and $\lambda \in \big((p-1)/pd, 1/d\big)$. There exists a constant $\theta > 0$ such that, for n large enough,*

$$\mathbb{P}\left(d_{GH}((\mathcal{M}, d_{f,p}), (\mathbb{X}_n, \tfrac{n^{(p-1)/d}}{\mu} d_{\mathbb{X}_n,p})) > \varepsilon\right) \leq \exp\left(-\theta n^{(1-\lambda d)/(d+2p)}\right).$$

3 Density-based manifold learning

In this section we couple the estimation of Fermat distance on input data with the Multidimensional Scaling method to achieve dimensionality reduction. This strategy is similar to the one used by the Isomap algorithm. However, it is known that the Isomap algorithm suffers from topological instability in presence of noise, since it may construct erroneous connections (called *short-circuits*) in the neighborhood graph that potentially impair its performance. In contrast, since noise generally corresponds with regions of low density, noisy points are treated by our method almost as not being part of the manifold. This effects increases with the value of p. In particular, they do not affect substantially the inference of the right geometry of the data.

▶ **Example 3.1** (Trefoil)**.** The Trefoil is a (non-trivial) knot, that is, a particular embedding of a topological circle $\mathbb{S}^1$ in $\mathbb{R}^3$. In particular, it is homeomorphic to $\mathbb{S}^1$. It is expected that a projection onto $\mathbb{R}^2$ should be a circle. However, in presence of noise, Isomap projection to $\mathbb{R}^2$ poorly recovers the underlying geometry of the data, while MDS with matrix distance computed with the sample estimator of Fermat metric globally recovers the existence of a circle (see Figure 2). The noise becomes isolated points as p increases.

4 Intrinsic persistence diagrams

In this section we apply Theorem 2.5 to the topological inference of features from data. In particular, we explore the convenience of the computation of persistence diagrams using as input the data endowed with the sample Fermat distance. We refer the reader to [2, 4, 5, 8] for a more complete exposition of the persistent homology theory.

For the computation of the persistent homology of a point cloud, one imagines each point as a *ball* (that is, representing a small surrounding region) and builds a combinatorial model for the space connecting the points according to whether the corresponding regions intersect. More precisely, for every fixed value of a parameter or *scale* that controls the size of the region that each point represents, one gets a *simplicial complex* (ie, a higher dimensional analogue of a graph). This family of simplicial complexes, also known as a *filtration*, is the input of the procedure to compute persistent homology. Indeed, the topological features of this family of complexes change as the scale parameter grows: different connected components join in one, some loops are filled, new cavities appear, etc. By analyzing these transitions, we are able to assign a *birth* and a *death* value to each of these features, and the difference between them represents its *lifetime*. The most persistent features represent *topological signatures*, whereas the shortest intervals may be considered as *noise*. The output of this procedure is summarized in an object called *persistence diagram*, denoted by $\mathrm{dgm}(\cdot)$ (see Figure 3).

In general, we usually only get an approximation of the metric space under consideration, so we will be interested in comparing persistence diagrams built on top different metric spaces. The *bottleneck distance*, denoted by $d_b(\cdot)$, is a frequently used quantity to measure

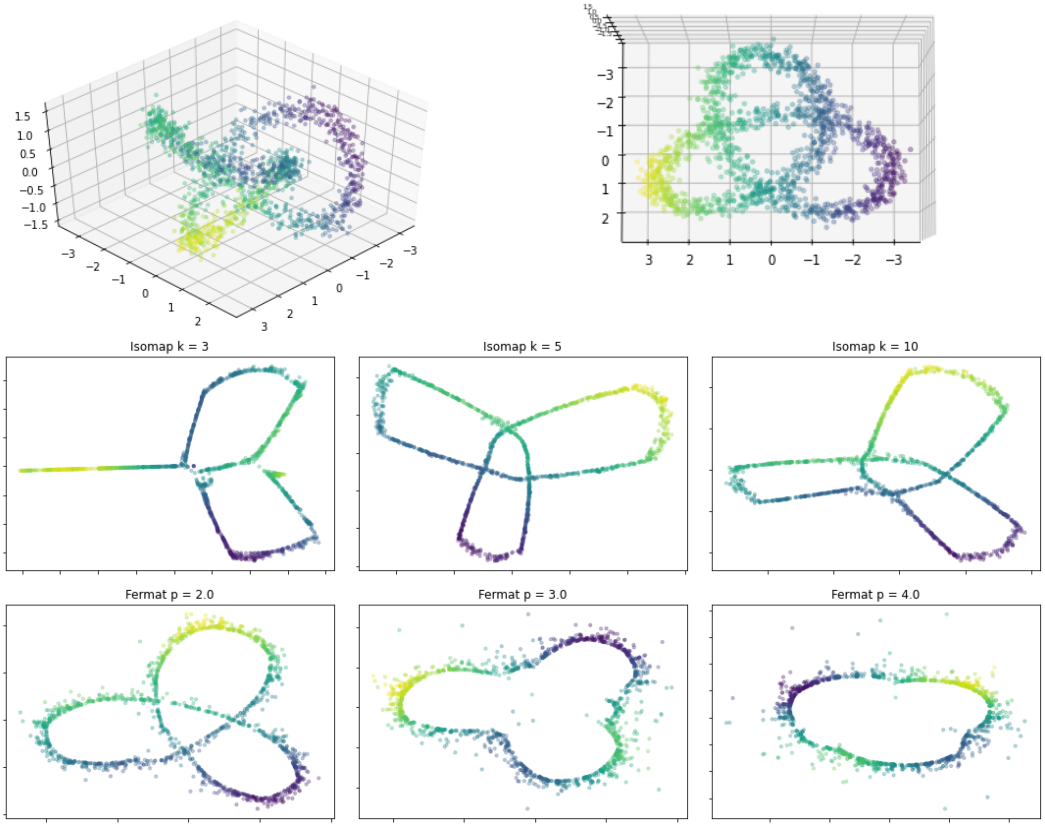


Figure 2 Reduction of dimensionality of the Trefoil knot using Isomap and Fermat distance + MDS.

the difference between two persistence diagrams. The stability theorem [5, 7] links this distance to the Gromov–Hausdorff distance between metric spaces. More concretely, it states that for any two precompact metric spaces $\mathbb{X}$ and $\mathbb{Y}$,

$$d_b\Big(\mathrm{dgm}\big(\mathbb{X},\rho_{\mathbb{X}}\big),\mathrm{dgm}\big(\mathbb{Y},\rho_{\mathbb{Y}}\big)\Big) \leq 2d_{GH}\big((\mathbb{X},\rho_{\mathbb{X}}),(\mathbb{Y},\rho_{\mathbb{Y}})\big),$$

where d_{GH} denotes the Gromov–Hausdorff distance.

We apply Theorem 2.5 to the estimation of the persistence diagram of a submanifold of an Euclidean space from a sample. This problem has already been studied in [6, 10], where the authors prove the almost sure convergence (in the sense of bottleneck distance) of the persistence diagrams associated to the sample to the persistence diagram of the desired metric space. However, in these works the distance function of the underlying metric space is assumed to be known which is not the case in most of real-life applications. We deduce an analogue result in our context, where the novelty lies in that both the underlying set and distance function of the metric space under study are unknown. This result is a direct consequence of Theorem 2.5 and the stability theorem.

▶ **Corollary 4.1.** *Let $\varepsilon > 0$ and $\lambda \in \big((p-1)/pd, 1/d\big)$. There exists a constant $\theta > 0$ such that, for n large enough,*

$$\mathbb{P}\Big(d_b\big(\mathrm{dgm}(\mathcal{M},d_{f,p}),\mathrm{dgm}(\mathbb{X}_n,\tfrac{n^{(p-1)/d}}{\mu}d_{\mathbb{X}_n,p})\big) > \varepsilon\Big) \leq \exp\left(-\theta n^{(1-\lambda d)/(d+2p)}\right).$$

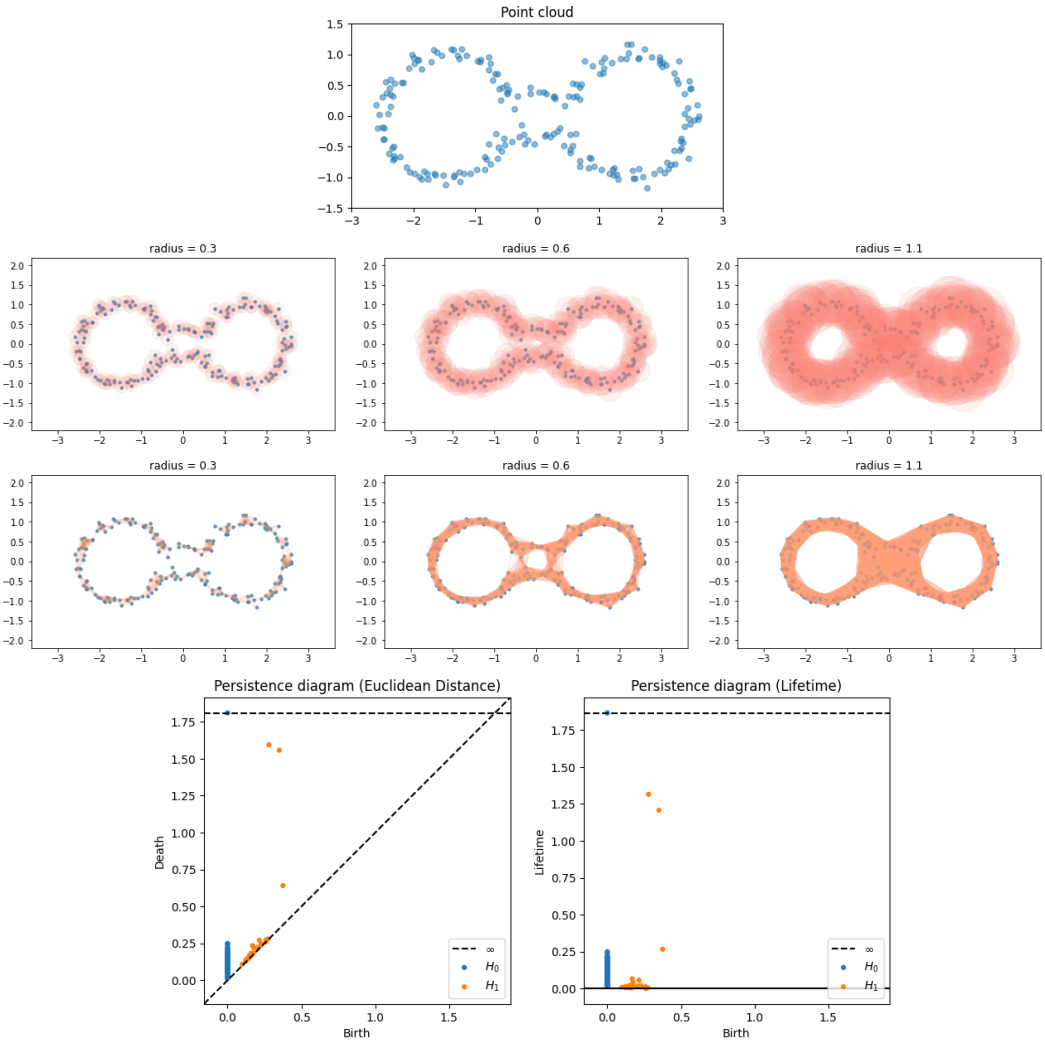


Figure 3 Persistent homology pipeline. From a point cloud construct a filtration of simplicial complexes parametrized by real numbers representing the radius of a covering of balls. Then, compute the persistent generators of homology groups and summarize the information in a persistence diagram.

▶ Remark. Persistent diagrams computed with Fermat distance and Euclidean distance are different in general. In persistence diagrams computed with intrinsic distances –rather than the Euclidean one– topological features persist longer. For instance, when Euclidean distance is used, the topology of the manifold is not expected to be reproduced for radii larger than the reach of $\mathcal{M}$. This limitation is absent when intrinsic metrics are used.

Persistence diagrams strongly depend on the notion of distance defined in the input data. In the next example we show the convenience of using Fermat distance.

▶ **Example 4.2** (Eyeglasses). We compute the persistence diagram associated to the sample points from Example 2.3, Figure 1. We compare the results obtained with different distance choices: the Euclidean distance, the Isomap estimator of the inherited Riemannian distance and the sample Fermat distance for $p = 2.5$ and $p = 3$ (see Figure 4). The homology of the eyeglasses curve has one generator of H_0 and one generator of H_1. However, it can be noticed that for both Euclidean and Isomap distances, the persistence diagram displays two

salient generators for the first homology group H_1, which can be attributed to the small reach of the manifold and the presence of noise. With the Fermat distance for different choices of p the diagram shows accurately only one persistent generator for H_1. On the other hand, the number of noticeable connected components increases with p. This effect is caused by the presence of noise in regions of extremely low density, becoming isolated points (or outliers) as p evolves.

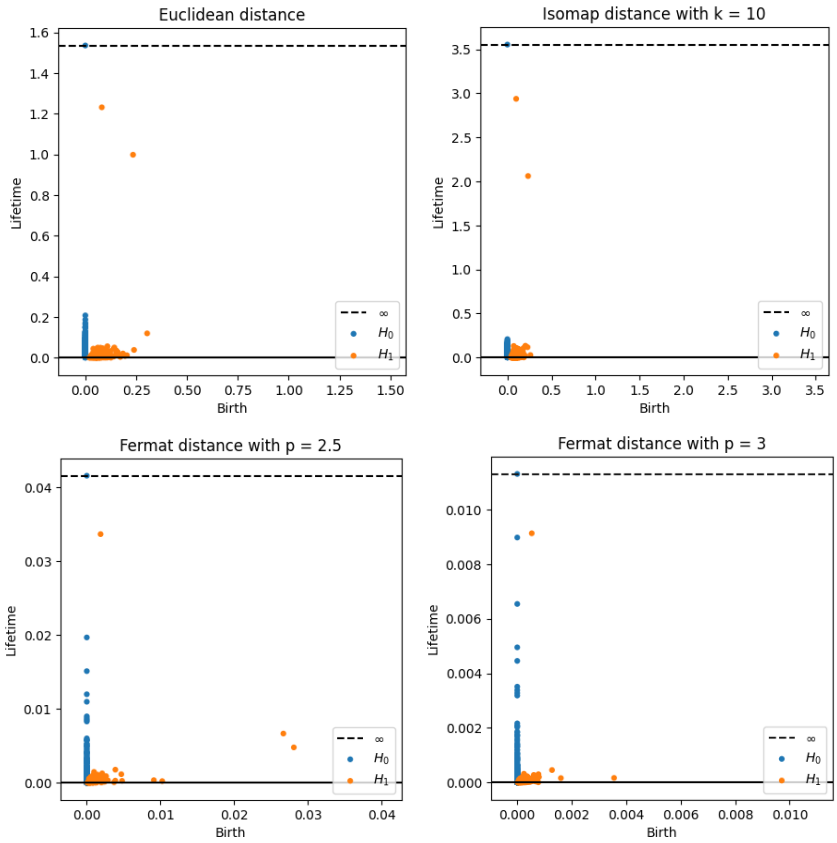


Figure 4 Persistence diagrams associated to the eyeglasses point cloud with noise for different distances: Euclidean, Isomap distance with $k = 10$, Fermat with $p = 2.5$ and $p = 3$.

References

1 M. Bernstein, V. De Silva, J. C. Langford, and J. B. Tenenbaum. Graph approximations to geodesics on embedded manifolds. Technical report, 2000.

2 Jean-Daniel Boissonnat, Frédéric Chazal, and Mariette Yvinec. *Geometric and topological inference*. Cambridge Texts in Applied Mathematics. Cambridge University Press, Cambridge, 2018.

3 Eugenio Borghini, Ximena Fernández, Pablo Groisman, and Gabriel Mindlin. Intrinsic persistent homology via density-based metric learning. *arXiv:2012.07621*, 2020.

4 Frédéric Chazal, Vin de Silva, Marc Glisse, and Steve Oudot. *The structure and stability of persistence modules*. SpringerBriefs in Mathematics. Springer, Cham, 2016.

5 Frédéric Chazal, Vin de Silva, and Steve Oudot. Persistence stability for geometric complexes. *Geom. Dedicata*, 173:193–214, 2014.

6 Frédéric Chazal, Marc Glisse, Catherine Labruère, and Bertrand Michel. Convergence rates for persistence diagram estimation in topological data analysis. *J. Mach. Learn. Res.*, 16:3603–3635, 2015.

7 David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. *Discrete Comput. Geom.*, 37(1):103–120, 2007.

8 Herbert Edelsbrunner and John Harer. Persistent homology—a survey. In *Surveys on discrete and computational geometry*, volume 453 of *Contemp. Math.*, pages 257–282. Amer. Math. Soc., Providence, RI, 2008.

9 Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. *Discrete Comput. Geom.*, 28(4):511–533, 2002. Discrete and computational geometry and graph drawing (Columbia, SC, 2001).

10 Brittany Terese Fasy, Fabrizio Lecci, Alessandro Rinaldo, Larry Wasserman, Sivaraman Balakrishnan, and Aarti Singh. Confidence sets for persistence diagrams. *Ann. Statist.*, 42(6):2301–2339, 2014.

11 Pablo Groisman, Matthieu Jonckheere, and Facundo Sapienza. Nonhomogeneous euclidean first-passage percolation and distance learning. *arXiv:1810.09398*, 2018.

12 C. Douglas Howard and Charles M. Newman. Euclidean models of first-passage percolation. *Probab. Theory Related Fields*, 108(2):153–170, 1997.

13 Sung Jin Hwang, Steven B. Damelin, and Alfred O. Hero, III. Shortest path through random points. *Ann. Appl. Probab.*, 26(5):2791–2823, 2016.

14 D. Mckenzie and S. Damelin. Power weighted shortest paths for clustering euclidean data. *arXiv:1905.13345*, 2019.

15 Partha Niyogi, Stephen Smale, and Shmuel Weinberger. Finding the homology of submanifolds with high confidence from random samples. *Discrete Comput. Geom.*, 39(1-3):419–441, 2008.

16 J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

17 W. S. Torgerson. Multidimensional scaling: I. theory and method. *Psychometrika*, 17:401–419, 1952.

18 Afra Zomorodian and Gunnar Carlsson. Computing persistent homology. *Discrete Comput. Geom.*, 33(2):249–274, 2005.

Bicolored Path Embedding Problems in Protein Folding Models

Tianfeng Feng[1], Ryuhei Uehara[2], and Giovanni Viglietta[3]

1 **School of Information Science, Japan Advanced Institute of Science and Technology (JAIST)**
ftflluy@jaist.ac.jp
2 **School of Information Science, Japan Advanced Institute of Science and Technology (JAIST)**
uehara@jaist.ac.jp
3 **School of Information Science, Japan Advanced Institute of Science and Technology (JAIST)**
johnny@jaist.ac.jp

Abstract

In this paper, we introduce a path embedding problem inspired by the well-known HP model of protein folding. A graph is said *bicolored* if each vertex is assigned a label in the set {red, blue}. For a given bicolored path P and a given bicolored graph G, our problem asks whether we can embed P into G in such a way as to match the colors of the vertices, or not.

We first show that our problem is NP-complete even if G is a dense graph of the same size as P. We then study the special case where G is a grid graph (a typical scenario in protein folding models), showing that the path embedding problem remains NP-complete even if P is monochromatic, or if G and P have the same size. By contrast, we prove that the path embedding problem becomes tractable if the grid graph G has fixed height. Finally, we show the NP-hardness of a maximization problem directly inspired by the HP model of protein folding.

1 Introduction

The protein folding problem asks how a protein's amino acid sequence dictates its three-dimensional atomic structure. This problem has wide applications and a long history dating back to the 1960s [7]. From the viewpoint of theoretical computer science, there is ongoing research aiming at revealing insights into reality by working on simplified abstract models.

One of the most popular such models is the *hydrophobic-polar* (HP) model [6, 8]. A protein in the HP model is represented as an abstract open chain, where each link has unit length and each joint is marked either H (hydrophobic, i.e., non-polar) or P (hydrophilic, i.e., polar). A protein is usually envisioned as a path embedded in a grid within the 2D or 3D lattice, where each joint in the chain maps to a point on the lattice, and each link maps to a single edge. The HP model of energy specifies that a chain desires to maximize the number of *H-H contacts*, which are pairs of H nodes that are adjacent on the lattice but not adjacent along the chain. The optimal folding problem in the HP model asks to find an embedding of a sequence of Hs and Ps on the 2D square lattice that maximizes the number of H-H contacts. This problem is known to be NP-hard in general [3].

Previous results on the HP model mostly concern the 2D square lattice, and some techniques rely on the properties of parity in a lattice (see [4, Sec. 9.3] for a comprehensive survey). However, such parity-related observations have no meaning in the original protein folding problem that we aim to model. Also, the number of H-H contacts is not the only possible measure that may be used to capture the intricate physical and chemical laws that

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

describe how a real protein folds. These facts have taken us to a new variant of the protein folding problem within the HP model, which we named *bicolored path embedding problem*.

In our model, we combine the basic ideas of protein folding with the complementary problem of *protein design*, where the goal is to synthesize a protein of a given shape (and function) from an amino acid sequence. Thus, we provide the "blueprint" of the folded shape of a protein, in the form of an input (grid) graph G with colors assigned to its vertices, and we ask if a given colored path P can be (injectively) embedded in G in such a way that vertex colors match. In other terms, we are effectively asking whether a given amino acid sequence can fold into (part of) a protein with prescribed structure. Since the HP model has nodes of only two types, we assume both G and P to be *bicolored*, say, with colors "red" and "blue".[1]

In Section 2, we prove that the bicolored path embedding problem is NP-complete even if G is a dense graph of the same *size* as P (i.e., with the same number of vertices).[2] In Section 3, we consider the case where G is a (square) grid graph, which is the standard assumption in the HP model. We first prove that the path embedding problem remains NP-complete even if P is monochromatic (e.g., all its vertices are blue), and then we prove that the problem is NP-complete even if G and P have the same size. Next, we contrast these hardness results with a polynomial-time algorithm for the case where G is a grid of fixed height: thus, the bicolored path embedding problem, parameterized according to the height of G, is in XP. In Section 4, we show that maximizing red-red contacts in the bicolored path embedding problem (defined in the same way as H-H contacts in the HP model) is also NP-hard.[3]

2 Bijective embedding in a dense graph

Let us first consider the case where the bicolored blueprint G is a "precise" description of a protein, i.e., it has to be matched exactly by the amino acid sequence represented by the bicolored path P. In other words, G and P have the same number of vertices, and the embedding should therefore be bijective. We will show that the embedding problem is NP-hard even if G is a dense graph (intuitively, a blueprint with many edges should allow greater leeway in the construction of an embedding of P) by a reduction from the strongly NP-complete *3-Partition* problem [9]. We recall that the input to the 3-Partition problem is a multiset of $3m$ positive integers $\{a_1, a_2, \ldots, a_{3m}\}$, and the goal is to decide whether it can be partitioned into m multisets of equal sum S. Our reduction is sketched in Figure 1.

The path P has length $m \cdot (S+1)$, and is made up of m consecutive copies of a sub-path denoted by P_{S+1}, which in turn consist of a red vertex followed by S blue vertices. The blueprint G contains a complete bipartite graph $K_{6m,m}$ with m red vertices on one side and $6m$ blue vertices on the other side. These blue vertices are further connected in all possible ways, forming a clique Q of size $6m$ (the gray box in the figure). Additionally, for each a_i, we construct a clique of $a_i - 2$ blue vertices (in the 3-Partition problem, we can safely assume that $a_i > 2$), and we connect two of its vertices with two vertices of Q.

---

[1] With regard to bicolored and monochromatic graphs, we do *not* adhere to established terminology from classical graph coloring theory; for the purposes of this paper, a coloring of a graph is simply a labeling of its vertices, with no extra constraints. In particular, adjacent vertices may have the same color.

[2] Formally, an infinite collection of graphs $\mathcal{S}$ is said to be a set of *dense graphs* if there is a positive constant α such that, for any large-enough n, every graph in $\mathcal{S}$ with n vertices has at least $\alpha \cdot n^2$ edges.

[3] We remark that, in previous work, it has been established that the problem of maximizing H-H contacts is NP-hard when G is not given, and P can be embedded in any way on a grid [3].

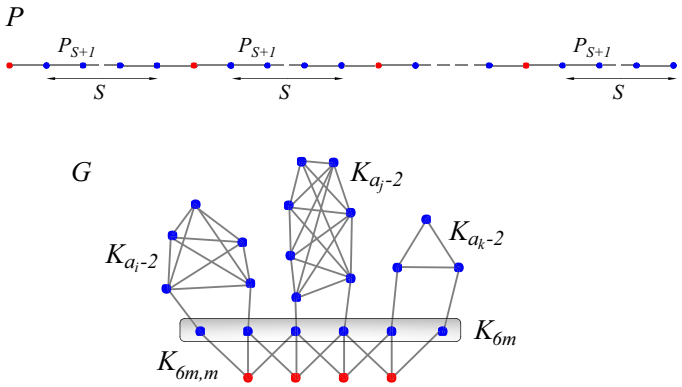


Figure 1 Sketch of the NP-hardness reduction from the 3-Partition problem. For clarity, the edges of the clique K_{6m}, as well as some edges of the complete bipartite graph $K_{6m,m}$, have been omitted.

It is easy to see that the graph G is dense, it has the same size as P, and there is an embedding of P into G if and only if the a_i's can be partitioned into multisets of sum S.

▶ **Theorem 1.** *The bicolored path embedding problem is NP-complete even if the blueprint G is a dense graph of the same size as the path P.* ◀

3 Embedding in a grid graph

In this section we focus on blueprint graphs G which are *grid graphs*, i.e., they are obtained from regular tilings of the plane (sometimes these are also called *lattice graphs*). This is the typical setting of the standard HP model.

3.1 Monochromatic path

If the path P only consists of blue vertices, there is a simple NP-hardness reduction from the *Hamiltonian path* problem (i.e., given a graph, decide if there is a walk that visits each vertex exactly once), which is known to be NP-complete even if the graph G is an induced subgraph of a square grid graph, a triangular grid graph, or a hexagonal grid graph [1, 2, 13].

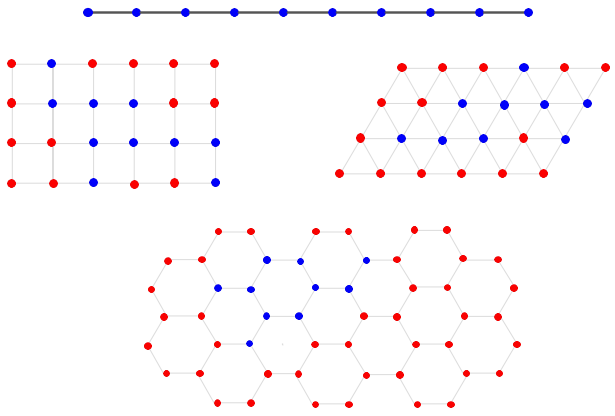


Figure 2 NP-hardness reduction from the Hamiltonian path problem for several grid graphs

Our reduction is sketched in Figure 2: given a graph G' on n vertices, which is an induced subgraph of a grid graph, we color all its vertices blue, and we "complete" it to a grid graph G by adding red vertices. Obviously, we can embed a path P of n blue vertices into G if and only if G' has a Hamiltonian path.

▶ **Theorem 2.** *The bicolored path embedding problem is NP-complete even if the blueprint G is a (square, triangular, or hexagonal) grid graph, and P is a monochromatic path.* ◀

3.2 Bijective embedding in a square grid graph

Let us turn again to bijective embeddings, this time in the case where the blueprint G is a square grid graph. We will give another NP-hardness reduction from the Hamiltonian path problem. We start from a square grid graph $R(m', n')$ with an induced subgraph G' (which is an instance of the Hamiltonian path problem), and we construct the blueprint G by "expanding" each vertex v of $R(m', n')$ into a $(k+2) \times (k+2)$ block B_v (where k is a large-enough even constant, defined later). If v is not a vertex of G', then all vertices of B_v are blue; if v is a vertex of G, then B_v is illustrated in Figure 3: its four central vertices are red, and all other vertices are blue. The size of G is therefore $(k+2) \cdot m' \times (k+2) \cdot n'$; the full construction is shown in Figure 4.

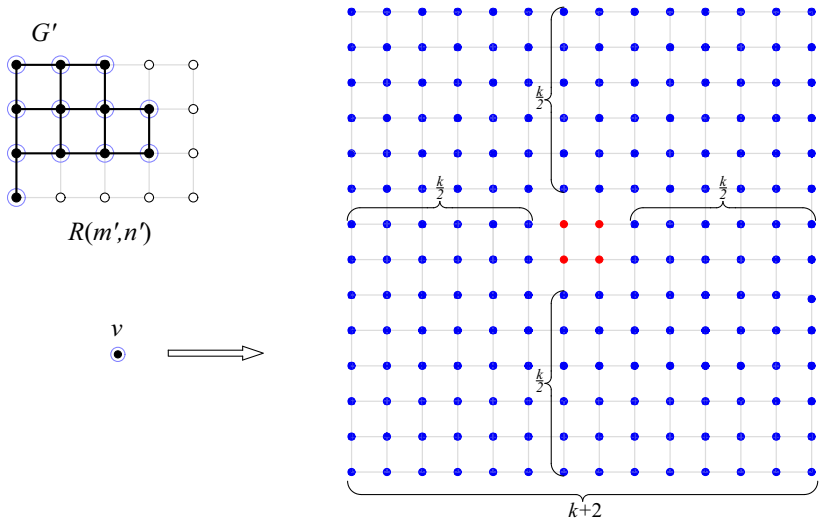


Figure 3 Transformation of a vertex v of G' into the $(k+2) \times (k+2)$ block B_v

The path P is sketched in Figure 5: there is a copy of the subpath P' for each vertex of G', and then a final trail of blue vertices such that the total length of P matches the size of G. Now, to embed P into G, we have to start from a set of four red vertices in some block B_v, and then move to another set of four red vertices in some other block B_w. Since we must traverse exactly $2k$ blue vertices between these two red sets, this is possible only if v and w are adjacent in G' (note that a "diagonal" move would take $2k+1$ steps on blue vertices). Thus, embedding P into G is impossible if G' is not Hamiltonian.

Assume now that G' is Hamiltonian. We can embed all copies of P' into G by "mimicking" a Hamiltonian path in G' and moving from one set of red vertices to the next by covering the $2 \times k$ rectangle between them in a zig-zag fashion. Eventually, the region of G covered by all the copies of P' looks like a winding "tube" of width 2, as sketched in Figure 6.

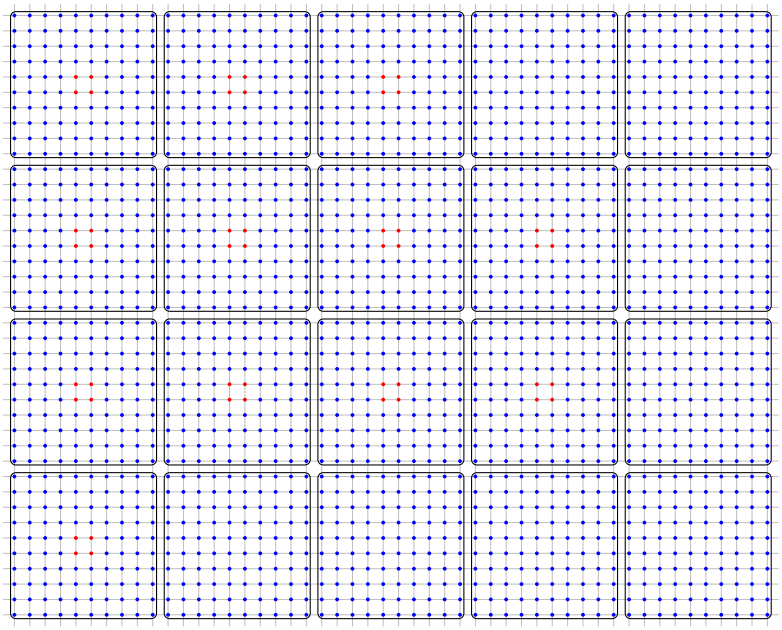


Figure 4 Complete construction of G: each block represents a vertex in the original graph

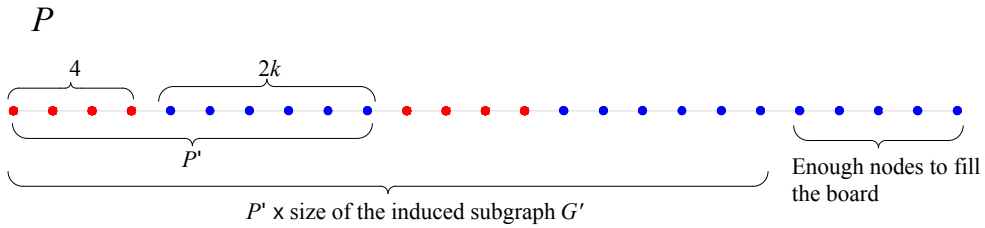


Figure 5 Construction of the path P

Now we have to cover the remaining part of G with the trailing sequence of blue vertices of P. In order to do that, we partition this region into maximal "horizontal rectangles", i.e., in such a way that no two rectangles touch each other along vertical edges, as shown in Figure 6. Then we do a depth-first traversal of these rectangles. When we visit a rectangle, we cover it as shown in Figure 7: we further divide it into smaller rectangular "tiles", one for each unvisited neighboring rectangle. After covering a tile, we visit its adjacent rectangle in the partition, and then we move to the next tile when we backtrack from that rectangle.

Constructing the tiles such that each of them can be covered completely before moving on to the next rectangle is indeed possible. In [10], the grid graphs containing a Hamiltonian path with assigned endpoints have been characterized: as it turns out, if the size of a tile is even and one of its sides is longer than four vertices, then there is a Hamiltonian path in the tile with any assigned endpoints having odd distance. Because k is a large even constant, we can indeed subdivide each rectangle in the appropriate number of tiles, each of which has even size and at least one side longer than four vertices (choosing $k = 100$ suffices by a big margin). It follows that we can embed P into G.

▶ **Theorem 3.** *The bicolored path embedding problem is NP-complete even if the blueprint G is a square grid graph of the same size as the path P.* ◀

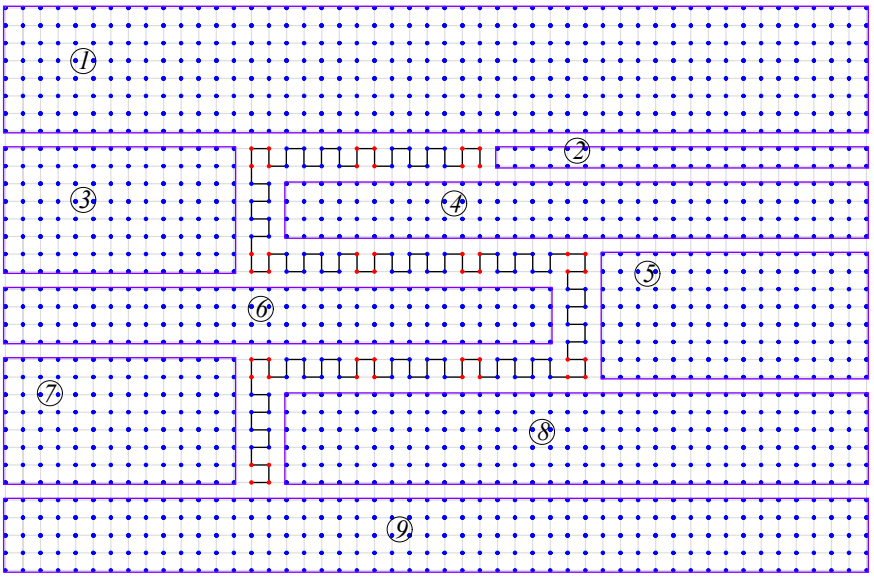


Figure 6 Partition into rectangles of the region not covered by the zig-zagging copies of P'

3.3 Fixed-height rectangular blueprint

We can contrast our previous hardness results with an embedding algorithm that runs in polynomial time, provided that the blueprint G is a grid graph of fixed height k. Thus, let G be a bicolored $m \times k$ grid, and let P be a bicolored path of n vertices. Our approach is based on dynamic programming, where a sub-problem consists of embedding part of P into a sub-grid of G going from the first column to the ath column, with $1 \leq a \leq m$. A sub-problem's specification also contains a description of the intersection between a hypothetical embedding of P and the ath column of G, illustrated in Figure 8: for each vertex w in the ath column, the sub-problem specifies which vertex v_i of P is mapped to w (if any), as well as an extra bit of information that encodes whether the left or right neighbor of v_i along P should be mapped to the left neighbor of w (if such information is incompatible with the rest of the specification, this bit is ignored). Thus, the total number of sub-problems is $2^k \cdot n^k \cdot m$ (the last factor represents the m choices of a).

The output to a sub-problem is "Yes" if an embedding satisfying the given constraints exists, "No" if it does not exist, and "N/A" if the sub-problem specifies no intersection on the ath column, and it is not possible to embed P entirely to the left of the ath colum (this implies that P should be embedded entirely to the right of the ath column, but we are still unable to determine if this is possible).

Solving a sub-problem S for column a amounts to finding a sub-problem S' for column $a-1$ with a "Yes" answer such that the specifications of S and S' are compatible. In other words, the mappings described by S and S' on columns a and $a-1$ should (i) match the colors in G and P, and (ii) match with each other: for example, if S indicates that the vertex v_{i+1} of P should be mapped to the left neighbor w' of w (where w is in column a), then S' should indicate that v_{i+1} is indeed mapped to w' (which is in column $a-1$). Thus, S can be solved by looking up at most n^k sub-problems, and each compatibility test takes $O(k)$ time.

▶ **Theorem 4.** *Given a bicolored square grid graph G of size $m \times k$ and a bicolored path P of size n, the embedding problem for G and P can be solved in $O(k \cdot 2^k \cdot n^{2k} \cdot m)$ time.* ◀

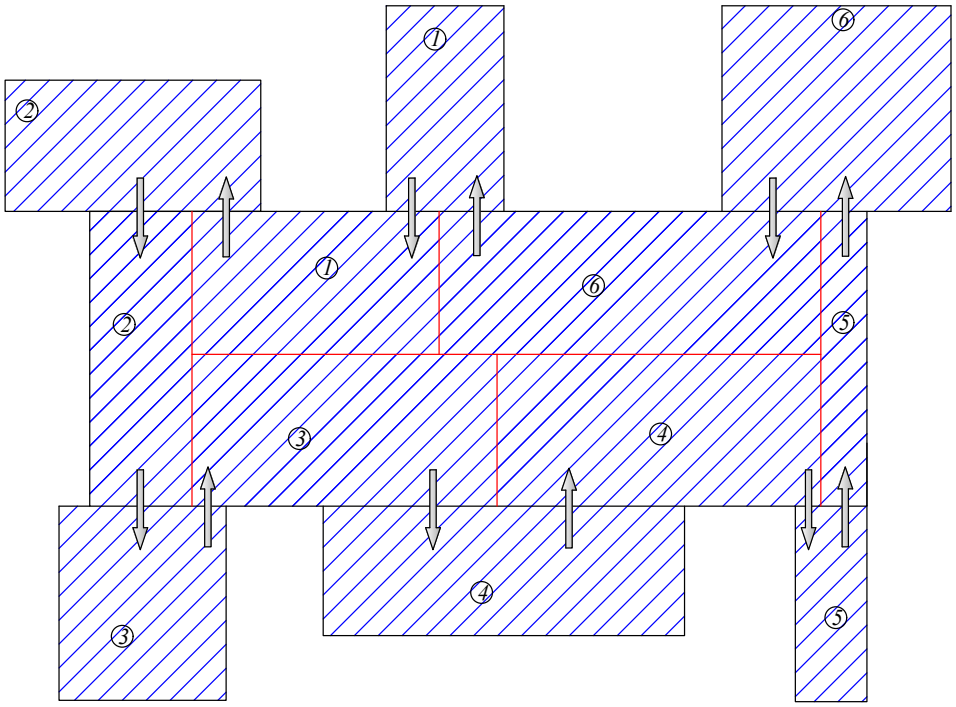


Figure 7 Traversal order of the tiles of a rectangle and its six neighboring rectangles

Note that, if k is a constant, the running time of our algorithm is $O(n^{2k}m)$, hence polynomial.

▶ **Corollary 5.** *The bicolored path embedding problem where the blueprint G is a square grid graph, parameterized according to the height of G, is in XP.* ◀

4 Maximizing red-red contacts in a grid graph

Finally, let us turn to the problem of maximizing red-red contacts in the context of the bicolored path embedding problem. Recall that, according to the HP model of energy, an amino acid chain tends to fold in a way that maximizes the number of H nodes that are close together in the folded state, even if they are not adjacent along the chain. In other words, when G and P are given, we seek an embedding of P into G that covers a large number of adjacent red vertices of G without traversing the edges between them. A red-red contact in an embedding of P is a pair of adjacent red vertices u, v in G such that the embedding of P covers both u and v, but does *not* contain the edge $\{u, v\}$.

The problem of maximizing red-red contacts in the bicolored path embedding problem is also NP-hard, even when restricted to instances where the path P is guaranteed to be embeddable into G. Figure 9 shows a reduction from the Hamiltonian path problem, where Block 2 of G is constructed as the graph in Section 3.1: that is, we are given a graph G', we color its n vertices blue, and then we "complete" it to a grid graph by adding r red vertices around it. Then we take an integer k greater than r, and we construct Block 1, which is a grid of at least k red vertices. Block 3 of G and the path P are constructed as in the figure.

Now it is easy to see that, if G' does not have a Hamiltonian path, we can only embed P in Block 3, which yields no red-red contacts. Otherwise, we can embed the blue part of P in Block 2 and the red part in Block 1, which produces a large number of red-red contacts.

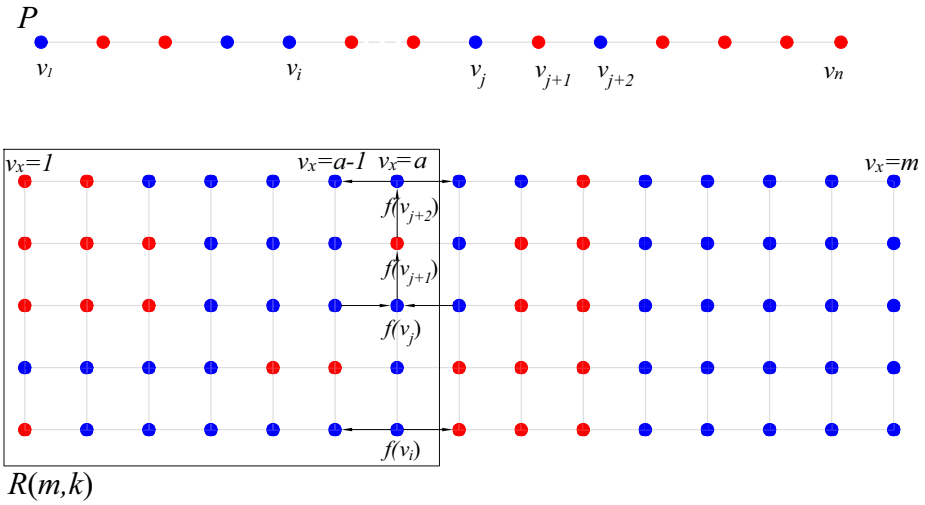


Figure 8 Illustration of the dynamic-programming algorithm for rectangular blueprints

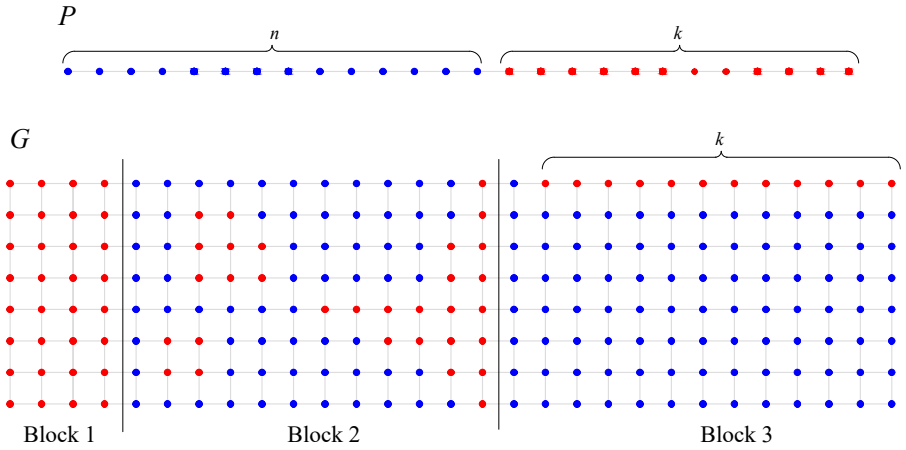


Figure 9 NP-hardness reduction for the problem of maximizing red-red contacts

▶ **Theorem 6.** *Given a bicolored grid graph G and a bicolored path P that can be embedded in G, it is NP-hard to find an embedding of P in G that maximizes red-red contacts.* ◀

This result can easily be extended to grid graphs induced by different tilings of the plane (cf. Section 3.1). Also, it shows that the related *approximation problem* is NP-hard, as well.

Acknowledgments. The authors wish to thank the anonymous reviewers for useful observations and suggestions. This work is partially supported by JSPS KAKENHI Grant Numbers 17H06287 and 18H04091.

References

1 Esther M. Arkin, Sándor P. Fekete, Kamrul Islam, Henk Meijer, Joseph S. B. Mitchell, Yurai Núñez-Rodríguez, Valentin Polishchuk, David Rappaport, and Henry Xiao. Not being (super) thin or solid is hard: A study of grid Hamiltonicity. *Computational Geometry*, 42(6–7):582–605, 2009.

2 Michael Buro. Simple Amazons endgames and their connection to Hamilton circuits in cubic subgrid graphs. In *International Conference on Computers and Games*, pages 250–261. Springer, 2000.

3 Pierluigi Crescenzi, Deborah Goldman, Christos Papadimitriou, Antonio Piccolboni, and Mihalis Yannakakis. On the complexity of protein folding. *Journal of Computational Biology*, 5(3):423–465, 1998.

4 Erik D. Demaine and Joseph O'Rourke. *Geometric folding algorithms: linkages, origami, polyhedra*. Cambridge University Press, 2007.

5 Erik D. Demaine and Mikhail Rudoy. Hamiltonicity is hard in thin or polygonal grid graphs, but easy in thin polygonal grid graphs. *arXiv:1706.10046*, 2017.

6 Ken A. Dill. Theory for the folding and stability of globular proteins. *Biochemistry*, 24(6):1501–1509, 1985.

7 Ken A. Dill and Justin L. MacCallum. The protein-folding problem, 50 yeas on. *Science*, 338(6110):1042–1046, 2012.

8 Kit Fun Lau and Ken A. Dill. A lattice statistical mechanics model of the conformational and sequence spaces of proteins. *Macromolecules*, 22(10):3986–3997, 1989.

9 Michael R. Garey and David S. Johnson. *Computers and intractability*, volume 174. Freeman San Francisco, 1979.

10 Alon Itai, Christos H. Papadimitriou, and Jayme Luiz Szwarcfiter. Hamilton paths in grid graphs. *SIAM Journal on Computing*, 11(4):676–686, 1982.

11 Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Springer, 1972.

12 F. Luccio and C. Mugnia. Hamiltonian paths on a rectangular chessboard. In *Proceedings of the 16th Annual Allerton Conference*, pages 161–173, 1978.

13 Christos H. Papadimitriou and Umesh V. Vazirani. On two geometric problems related to the travelling salesman problem. *Journal of Algorithms*, 5(2):231–246, 1984.

On Voronoi diagrams of 1.5D terrains with multiple viewpoints*

Vahideh Keikha[1] and Maria Saumell[1,2]

1 **The Czech Academy of Sciences, Institute of Computer Science, Czech Republic**
(keikha,saumell)@cs.cas.cz
2 **Department of Theoretical Computer Science, Faculty of Information Technology, Czech Technical University in Prague, Czech Republic**

Abstract

Given an n-vertex 1.5D terrain $\mathcal{T}$ and a set $\mathcal{P}$ of $m < n$ viewpoints, the Voronoi visibility map $\mathrm{VorVis}(\mathcal{T},\mathcal{P})$ is a partitioning of $\mathcal{T}$ into regions such that each region is assigned to the closest visible viewpoint. The colored visibility map $\mathrm{ColVis}(\mathcal{T},\mathcal{P})$ is a partitioning of $\mathcal{T}$ into regions that have the same set of visible viewpoints. In this paper, we propose an algorithm to compute $\mathrm{VorVis}(\mathcal{T},\mathcal{P})$ which runs in $O(n+(m^2+k_c)\log n)$ time, where k_c and k_v denote the total complexity of $\mathrm{ColVis}(\mathcal{T},\mathcal{P})$ and $\mathrm{VorVis}(\mathcal{T},\mathcal{P})$, respectively. This improves upon a previous algorithm for this problem. We also show that the next problem can be solved in the same running time: What is the minimum value of r such that, if the viewpoints can only see objects within distance r, the partitioning of $\mathcal{T}$ into visible and invisible portions does not change?

1 Introduction

A 1.5D terrain $\mathcal{T}$ is an x-monotone polygonal chain of n vertices in $\mathbb{R}^2$. Two points on $\mathcal{T}$ are *visible* if the segment connecting them does not contain any point strictly below $\mathcal{T}$.

Visibility problems in terrains are fundamental in geographical information science and have many applications, such as placing fireguard or telecommunication towers [3], identifying areas that are not visible from sensitive sites [14], or solving problems related to sensor networks [16]. Although 2.5D terrains are more interesting for modelling and forecasting, 1.5D terrains are easier to visualize and they give insights into the difficulties of 2.5D terrains in terrain analysis. We focus on the variant where a set $\mathcal{P}$ of m viewpoints are located on vertices of $\mathcal{T}$, and our goal is to efficiently extract information about the visibility of $\mathcal{T}$ with respect to $\mathcal{P}$. We continue the work of [11], where the following structures are defined.

The *visibility map* $\mathrm{Vis}(\mathcal{T},\mathcal{P})$ is a partitioning of $\mathcal{T}$ into a *visible* region (containing all portions of $\mathcal{T}$ that are visible by at least one element in $\mathcal{P}$) and an *invisible* region (containing the portions that are not visible by any element in $\mathcal{P}$). The *colored visibility map* $\mathrm{ColVis}(\mathcal{T},\mathcal{P})$ is a partitioning of $\mathcal{T}$ into regions that have the same set of visible viewpoints (see Fig. 1b for an example). Finally, the *Voronoi visibility map* $\mathrm{VorVis}(\mathcal{T},\mathcal{P})$ is a partitioning of $\mathcal{T}$ into regions that have the same closest visible viewpoint (see Fig. 1c), where the distance used is the Euclidean distance (not the distance along the terrain).

Algorithms to compute these structures for 1.5D and 2.5D terrains are proposed in [11]. The algorithm to obtain $\mathrm{VorVis}(\mathcal{T},\mathcal{P})$ of a 1.5D terrain runs in $O(n+(m^2+k_c)\log n + k_v(m+\log n\log m))$ time, where k_c and k_v denote the total complexity of $\mathrm{ColVis}(\mathcal{T},\mathcal{P})$ and

* Supported by the Czech Science Foundation, grant number GJ19-06792Y, and with institutional support RVO:67985807. This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 734922.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

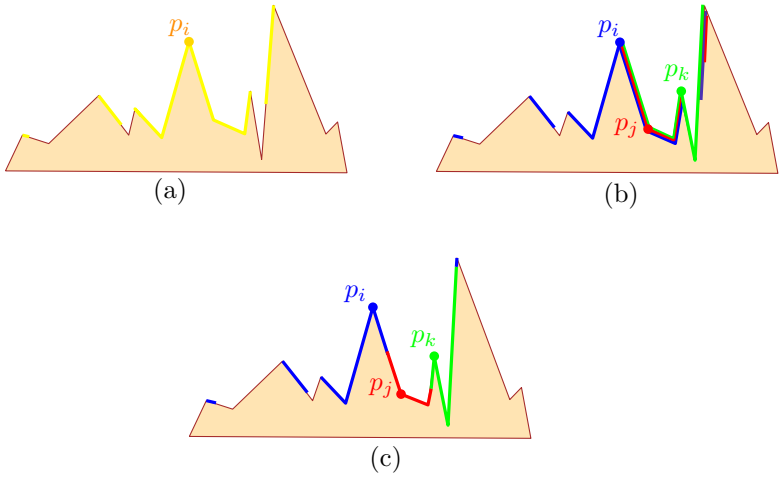


Figure 1 (a) The viewshed of a viewpoint. (b) ColVis($\mathcal{T}, \mathcal{P}$). (c) VorVis($\mathcal{T}, \mathcal{P}$).

VorVis($\mathcal{T}, \mathcal{P}$), respectively. It first computes ColVis($\mathcal{T}, \mathcal{P}$), and then it spends $\Theta(m)$ time to find each single region of VorVis($\mathcal{T}, \mathcal{P}$). In this note, we show that VorVis($\mathcal{T}, \mathcal{P}$) can be extracted from ColVis($\mathcal{T}, \mathcal{P}$) in a 1.5D terrain without adding any extra running time. We use an observation related to the bisectors of pairs of viewpoints that also allows to prove a relation between k_c and k_v.

The new algorithm for VorVis($\mathcal{T}, \mathcal{P}$) also allows to efficiently solve a problem related to limited range of sight. These problems are motivated by the fact that, even though many visibility problems assume an infinite range of visibility, the intensity of light, signals... decreases over distance in realistic environments. In this spirit, the problem of illuminating a polygonal area with the minimum total energy was introduced by O'Rourke [15], and studied in [6, 7]. We consider a related problem on terrains, namely, computing the minimum value r^* such that, if the viewpoints can only see objects within distance r^*, the obtained visibility map is the same as Vis($\mathcal{T}, \mathcal{P}$). We show that this problem can also be solved in $O(n+(m^2+k_c)\log n)$ time.

Related Work. When $m = 1$, computing the visibility map of a 1.5D terrain can be done in $O(n)$ time [12]. One of the first results on the variant where $m > 1$ is an $O((n+m)\log m)$ time algorithm to detect if there are any visible pairs of viewpoints above a 1.5D terrain [2]. Later, a systematic study of Vis($\mathcal{T}, \mathcal{P}$), VorVis($\mathcal{T}, \mathcal{P}$) and ColVis($\mathcal{T}, \mathcal{P}$) was carried out in [11] for 1.5D and 2.5D terrains. Apart from the mentioned output-sensitive algorithm for VorVis($\mathcal{T}, \mathcal{P}$) of a 1.5D terrain, the authors also propose an algorithm running in $O(mn \log m)$ time, which is worst-case nearly optimal, since the maximum complexity of VorVis($\mathcal{T}, \mathcal{P}$) is $\Theta(mn)$. A problem which is very related to the construction of Vis($\mathcal{T}, \mathcal{P}$) is that of computing the total visibility index of the terrain, that is, the number of viewpoints that are visible from each of the viewpoints. This problem can be solved in $O(n \log^2 n)$ time [1].

The situation where the locations of the viewpoints are unknown has been much studied. It is well-known that computing the minimum number of viewpoints to keep a 1.5D terrain illuminated is NP-Hard [13], but the problem admits a PTAS [8, 9, 10]. If the viewpoints are restricted to lie on a line, the same problem can be solved in linear time [5].

Assumptions. As in [11], we assume that no three vertices of $\mathcal{T}$ are aligned. Here we also assume that no edge of $\mathcal{T}$ is contained in the bisector of two viewpoints in $\mathcal{P}$.

Omitted proofs and details will be given in the full version of this paper.

2 Complexity of the Voronoi visibility map

In this section, we prove an upper bound on k_v.

Let us introduce some terminology. The *viewshed* of a viewpoint p is the set of points of $\mathcal{T}$ that are visible from p (see Fig. 1a for an example). Further, the *Voronoi viewshed* $\mathcal{W}_{\mathcal{T}}(p, \mathcal{P})$ of p is the set of points in the viewshed of p that are closer to p than to any other viewpoint that is visible from them.

We denote by $b_{i,j}$ the perpendicular bisector of two viewpoints p_i, p_j. Since no edge of $\mathcal{T}$ is contained in the bisector of two viewpoints, the shared boundary between two consecutive regions of VorVis$(\mathcal{T}, \mathcal{P})$ is a single point of $\mathcal{T}$, which we call an *event* point of VorVis$(\mathcal{T}, \mathcal{P})$.

We denote by $q_{i,j}$ an event point of VorVis$(\mathcal{T}, \mathcal{P})$ such that a point infinitesimally to the left and right of $q_{i,j}$ belongs to $\mathcal{W}_{\mathcal{T}}(p_i, \mathcal{P})$ and $\mathcal{W}_{\mathcal{T}}(p_j, \mathcal{P})$, respectively (notice that an event $q_{i,j}$ is different from an event $q_{j,i}$). There are three (not mutually exclusive) possibilities: (i) p_i becomes invisible at $q_{i,j}$[1]; (ii) p_j becomes visible at $q_{i,j}$; (iii) p_i and p_j are visible at $q_{i,j}$, and $q_{i,j}$ is an intersection point between $b_{i,j}$ and $\mathcal{T}$.

▶ **Lemma 2.1.** *Let $p_i \in \mathcal{P}$ be lower than $p_j \in \mathcal{P}$. Let q be an intersection between $b_{i,j}$ and $\mathcal{T}$ to the left (resp. right) of p_i. Then any point to the left (resp. right) of q visible from p_i is closer to p_j than to p_i. Thus, there is no event $q_{i,j}$ or $q_{j,i}$ of type (iii) to the left (resp. right) of q.*

We can now prove the following:

▶ **Theorem 2.2.** $k_v \leq k_c + m^2$.

Proof. Notice that events of type (i) and (ii) are also events of ColVis$(\mathcal{T}, \mathcal{P})$. Let us prove that there are at most m^2 events of type (iii): Let p_i, p_j be a pair of viewpoints. If p_i and p_j are at the same height, $b_{i,j}$ is vertical and only intersects $\mathcal{T}$ once, so there is at most one event of VorVis$(\mathcal{T}, \mathcal{P})$ on $b_{i,j} \cap \mathcal{T}$. Otherwise, we assume without loss of generality that p_i is lower than p_j. By Lemma 2.1 the only candidates for events $q_{i,j}$ or $q_{j,i}$ of type (iii) are the left-most intersection point of type $b_{i,j} \cap \mathcal{T}$ among all such points to the right of p_i and the right-most one among all points to the left. Thus, every pair of viewpoints creates at most two events of type (iii). ◀

3 Computation of the Voronoi visibility map

Let VorVis$(\mathcal{T}, \mathcal{P}_\ell)$ and ColVis$(\mathcal{T}, \mathcal{P}_\ell)$ be the corresponding maps if viewpoints only see themselves and to their left. VorVis$(\mathcal{T}, \mathcal{P}_r)$ and ColVis$(\mathcal{T}, \mathcal{P}_r)$ are defined analogously.

The outline of the algorithm we propose is given in Figure 2. Its main highlight is a way to compute VorVis$(\mathcal{T}, \mathcal{P}_\ell)$ and VorVis$(\mathcal{T}, \mathcal{P}_r)$ so that new events of the diagrams are found in $O(\log m)$ time rather than $O(m)$. We next explain the algorithm to compute VorVis$(\mathcal{T}, \mathcal{P}_r)$.

The algorithm sweeps the terrain from left to right and stops at points that are candidates for event points. The candidates for events of type (i) and (ii) are the events of ColVis$(\mathcal{T}, \mathcal{P}_r)$. We explain below which are the candidates for events of type (iii).

[1] When we write that p_i becomes invisible at $q_{i,j}$, we mean that it is visible immediately to the left of $q_{i,j}$ and invisible immediately to its right. We use the same rational when we write that p_j becomes invisible at $q_{i,j}$.

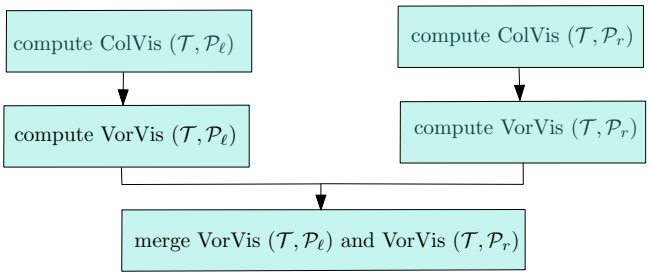


Figure 2 Outline of the algorithm to compute VorVis$(\mathcal{T}, \mathcal{P})$.

Events of ColVis$(\mathcal{T}, \mathcal{P}_r)$**.** We compute ColVis$(\mathcal{T}, \mathcal{P}_r)$ using the algorithm from [11] which returns a doubly-linked list with the vertices of ColVis$(\mathcal{T}, \mathcal{P}_r)$ sorted from left to right, together with the following visibility information: The visible viewpoints are specified for the first component of ColVis$(\mathcal{T}, \mathcal{P}_r)$ and, for the other components, the algorithm outputs the changes in the set of visible viewpoints with respect to the component immediately to the left.

Data structures. A binary table V indicates, at any given point of the sweep, which are the visible viewpoints, that is, $V[i] = 1$ if and only if p_i is visible at the moment.

Maintaining the set of viewpoints sorted by distance to the point of $\mathcal{T}$ currently swept by the line would be too expensive because every bisector $b_{i,j}$ might intersect the terrain $\Theta(n)$ times. Instead, we maintain a table L containing all viewpoints that have been swept by the line such that some pairs of viewpoints are sorted with respect to the distance to the current intersection point between the sweep line and the terrain, but some other pairs (which are never checked by the algorithm) are unordered. The table is filled by the algorithm from the last position to the first.

Additionally, we maintain a balanced binary search tree H which contains the positions of L occupied by viewpoints that are currently visible. As we will see, the viewpoints that are currently visible appear in L correctly sorted with respect to the current distance to the terrain. The algorithm always chooses as the "owner" of the current Voronoi visibility region the following viewpoint: among all elements that are currently visible, the one appearing earliest in L, whose position in L is found by obtaining the minimum element of H.

For all i, we also maintain a pointer from $V[i]$ to the position in L (if any) containing p_i.

Finally, we also use a data structure that allows to answer ray-shooting queries in $\mathcal{T}$ in $O(\log n)$ time [4]

Unordered sets of viewpoints and candidates for events of type (iii). Let $p_i \in \mathcal{P}$ be lower than $p_j \in \mathcal{P}$. Let q be the left-most intersection point of type $b_{i,j} \cap \mathcal{T}$ among all such points to the right of p_i. If p_j is to the left of p_i, q is to the right of both p_i and p_j, while if p_j is to the right of p_i, q lies between both viewpoints. In both cases, a point infinitesimally to the left of q is closer to p_i than to p_j, and a point infinitesimally to the right is closer to p_j.

If p_j is to the left of p_i, by Lemma 2.1, q is the only candidate for event of type (iii) to the right of p_i. Such a point is found in $O(\log n)$ time by a ray-shooting query and added to the list of candidates for events swept by the line. When the line sweeps q, p_i and p_j are swapped in L because p_j becomes closer to the terrain. Even though later the terrain might intersect $b_{i,j}$ again and p_i might become closer than p_j, p_i and p_j are *never* swapped again in L. The reason for this is that the algorithm only takes into account the visible viewpoints to output the Voronoi visibility regions and, by Lemma 2.1, any point to the right of q which

is visible from p_i is closer to p_j than to p_i. Therefore, whenever both p_i and p_j are visible, p_j is indeed closer to the terrain than p_i.

If p_j is to the right of p_i, since q is to the left of p_j and viewpoints can only see to their right, q is not a candidate for event of type (iii). By Lemma 2.1, there is no candidate for event of type (iii) to the right of p_i.

Finally, notice that there are no candidates for events of type (iii) to the left of p_i because in $\mathrm{VorVis}(\mathcal{T}, \mathcal{P}_r)$ viewpoints can only see to their right.

Description of the algorithm. Given q, r on $\mathcal{T}$ with $x(q) < x(r)$, we denote by $\mathcal{T}[q, r]$ the closed portion of the terrain between q and r. We create a list E of potential events sorted from left to right containing all events of $\mathrm{ColVis}(\mathcal{T}, \mathcal{P}_r)$ and the $O(m^2)$ candidates for events of type (iii). We also add an event at the right-most point of the terrain. Using E and the data structures mentioned above, we output $\mathrm{VorVis}(\mathcal{T}, \mathcal{P}_r)$ as a list of pairs $([q, r], p_i)$ such that p_i is the closest visible viewpoint in $\mathcal{T}[q, r]$ (if $\mathcal{T}[q, r]$ is not visible from any viewpoint, we output $([q, r], \perp)$). The variables t_ℓ and p_* in the algorithm below refer to the left endpoint of the portion of $\mathcal{T}$ currently analyzed by the algorithm and the closest visible viewpoint in that portion, respectively. The variable $p_{\min}$ refers to the viewpoint contained in the position of L given by the minimum element of H (if H is empty, $p_{\min} = \perp$).

Initially, $V[i] := 0$ and $L[i] := \emptyset$ for all i, $H := \emptyset$, $t_\ell :=$ left-most point of $\mathcal{T}$, and $p_* \leftarrow \perp$.

We repeat the following procedure until E is empty: We extract the next element q from E, and proceed according to the following cases (for the sake of simplicity, in the description below we deliberately ignore some degenerate situations which we tackle right after):

(i) One or more viewpoints become visible at q. For all such viewpoints p_i, we set $V[i] := 1$. If one of these viewpoints is at q, we insert it at the highest index j such that $L[j] = \emptyset$. For every viewpoint becoming visible at q, we insert to H the position of L containing it. If, after updating $p_{\min}$, $p_{\min} \neq p_*$, we output $([t_\ell, q], p_*)$, set $t_\ell := q$, and set $p_* := p_{\min}$.

(ii) One or more viewpoints become invisible at q. For all such viewpoints p_i, we set $V[i] := 0$ and we delete from H the position of L containing p_i. If, after updating $p_{\min}$, $p_{\min} \neq p_*$, we output $([t_\ell, q], p_*)$, set $t_\ell := q$, and set $p_* := p_{\min}$.

(iii) q is an intersection point between $\mathcal{T}$ and $b_{i,j}$. We swap p_i and p_j in L. If necessary, we update H according to the visibility of the viewpoints contained in the positions of L affected by the swap (for example, if p_i is visible and p_j is not, we add to H the new position of L containing p_i and we remove the old one). If, after updating $p_{\min}$, $p_{\min} \neq p_*$, we output $([t_\ell, q], p_*)$, set $t_\ell := q$, and set $p_* := p_{\min}$. If q is an intersection point between $\mathcal{T}$ and other bisectors distinct from $b_{i,j}$, the bisectors can be processed in any order.

(iv) q is the right-most point of $\mathcal{T}$. We output $([t_\ell, q], p_*)$.

It remains to explain in which order to process events of distinct type occurring at the same time. Viewpoints becoming visible have priority over the other types of events, and viewpoints becoming invisible have priority over intersections of the terrain with bisectors.

Merging the two diagrams. We overlap $\mathrm{VorVis}(\mathcal{T}, \mathcal{P}_\ell)$ and $\mathrm{VorVis}(\mathcal{T}, \mathcal{P}_r)$, and decompose the terrain by sweeping it from left to right and stopping at the endpoint of every pair of the form $([q, r], p_i)$ belonging to any of the two diagrams. After this decomposition, for any portion of the terrain there are two candidate viewpoints, and the distribution of the portion between the two viewpoints can be done in $O(\log n)$ time via a ray shooting query with their bisector (with similar arguments to those in the proof of Lemma 2.1, one can prove that the portion gets split into at most two portions of the final diagram).

We conclude:

▶ **Theorem 3.1.** VorVis$(\mathcal{T},\mathcal{P})$ *can be constructed in* $O(n+(m^2+k_c)\log n)$ *time and* $O(n+m^2+k_c)$ *space.*

4 Computation of r^*

Notice that r^* is realized at a vertex of VorVis$(\mathcal{T},\mathcal{P})$ at maximum distance from its closest visible viewpoint (r^* is the distance from the vertex to the viewpoint). Thus, after constructing VorVis$(\mathcal{T},\mathcal{P})$, we can traverse it in linear time to identify such a vertex. We obtain:

▶ **Theorem 4.1.** *The problem of computing the minimum range of the viewpoints that keeps* Vis$(\mathcal{T},\mathcal{P})$ *unchanged can be solved in* $O(n+(m^2+k_c)\log n)$ *time.*

5 Final remark

We have shown that VorVis$(\mathcal{T},\mathcal{P})$ can be constructed almost for free (asymptotically) if ColVis$(\mathcal{T},\mathcal{P})$ is computed first. Therefore, any faster algorithm for VorVis$(\mathcal{T},\mathcal{P})$ must use less information than the one encoded in ColVis$(\mathcal{T},\mathcal{P})$.

References

1. Peyman Afshani, Mark De Berg, Henri Casanova, Ben Karsin, Colin Lambrechts, Nodari Sitchinava, and Constantinos Tsirogiannis. An efficient algorithm for the 1D total visibility-index problem and its parallelization. *Journal of Experimental Algorithmics*, 23:1–23, 2018.
2. Boaz Ben-Moshe, Olaf Hall-Holt, Matthew J Katz, and Joseph SB Mitchell. Computing the visibility graph of points within a polygon. In *Proceedings of the 20th Annual Symposium on Computational Geometry*, pages 27–35, 2004.
3. Filipe X. Catry, Francisco C. Rego, Teresa Santos, Joel Almeida, and Paulo Relvas. Forest fires prevention in Portugal - using GIS to help improving early fire detection effectiveness. In *Proceedings of the 4th International Wildland Fire Conference*, 2007.
4. Bernard Chazelle, Herbert Edelsbrunner, Michelangelo Grigni, Leonidas J. Guibas, John Hershberger, Micha Sharir, and Jack Snoeyink. Ray shooting in polygons using geodesic triangulations. *Algorithmica*, 12(1):54–68, 1994.
5. Ovidiu Daescu, Stephan Friedrichs, Hemant Malik, Valentin Polishchuk, and Christiane Schmidt. Altitude terrain guarding and guarding uni-monotone polygons. *Computational Geometry: Theory and Applications*, 84:22–35, 2019.
6. Friedrich Eisenbrand, Stefan Funke, Andreas Karrenbauer, and Domagoj Matijevic. Energy-aware stage illumination. *International Journal of Computational Geometry & Applications*, 18(01n02):107–129, 2008.
7. Maximilian Ernestus, Stephan Friedrichs, Michael Hemmer, Jan Kokemüller, Alexander Kröller, Mahdi Moeini, and Christiane Schmidt. Algorithms for art gallery illumination. *Journal of Global Optimization*, 68(1):23–45, 2017.
8. Stephan Friedrichs, Michael Hemmer, James King, and Christiane Schmidt. The continuous 1.5D terrain guarding problem: Discretization, optimal solutions, and PTAS. *Journal of Computational Geometry*, 7(1):256–284, 2016.
9. Stephan Friedrichs, Michael Hemmer, and Christiane Schmidt. A PTAS for the continuous 1.5D terrain guarding problem. In *Proceedings of the 26th Canadian Conference on Computational Geometry*, 2014.
10. Matt Gibson, Gaurav Kanade, Erik Krohn, and Kasturi Varadarajan. An approximation scheme for terrain guarding. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 140–148. Springer, 2009.

11 Ferran Hurtado, Maarten Löffler, Inês Matos, Vera Sacristán, Maria Saumell, Rodrigo I. Silveira, and Frank Staals. Terrain visibility with multiple viewpoints. *International Journal of Computational Geometry & Applications*, 24(04):275–306, 2014.

12 Barry Joe and Richard B Simpson. Corrections to Lee's visibility polygon algorithm. *BIT Numerical Mathematics*, 27(4):458–473, 1987.

13 James King and Erik Krohn. Terrain guarding is NP-hard. *SIAM Journal on Computing*, 40(5):1316–1339, 2011.

14 Bernd Möller. Changing wind-power landscapes: regional assessment of visual impact on land use and population in Northern Jutland, Denmark. *Applied Energy*, 83(5):477–494, 2006. doi:10.1016/j.apenergy.2005.04.004.

15 Joseph O'Rourke. Open problems from CCCG 2005. In *Proceedings of the 18th Canadian Conference on Computational Geometry*, 2006.

16 Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. Wireless sensor network survey. *Computer Networks*, 52(12):2292–2330, 2008. doi:10.1016/j.comnet.2008.04.002.

Upward Planar Drawings with Three Slopes

Jonathan Klawitter and Johannes Zink

Universität Würzburg, Germany, {firstname.lastname}@uni-wuerzburg.de

Abstract

We study upward planar straight-line drawings that use only three different slopes. We show that deciding whether a digraph admits such a drawing is NP-hard already for embedded outerplanar digraphs, though linear-time solvable for trees with and without given embedding.

1 Introduction

One of the main goals in graph drawing is to generate clear drawings. To achieve this, we may reduce the visual complexity by using only few different geometric primitives [22] – in our case few slopes. Using only few different slopes is common for schematic drawings; for example, if we allow two different slopes we get orthogonal drawings [9], with three or four slopes we get hexalinear and octilinear drawings [31], respectively. Here, we additionally require our graphs to be drawn upward and planar.

Upward planarity. An *upward planar drawing* of a directed graph (or digraph for short) G is a planar drawing of G where every edge is drawn as a monotonic upward curve. We call G *upward planar* if it admits an upward planar drawing and *upward plane* if it is equipped with an upward planar embedding. Note that an upward planar embedding, given by the cyclic sequences of edges around each vertex, is necessarily bimodal, that is, each cyclic sequence can be split into a contiguous subsequence of incoming edges and a contiguous subsequence of outgoing edges [9].

Upward planarity testing is an NP-complete problem for general digraphs [17], though several fixed-parameter tractable algorithms exist [5, 14, 18]. Furthermore, the problem is polynomial-time solvable, for example for single source digraphs [3], outerplanar digraphs [33], series-parallel digraphs [14], and triconnected digraphs [2]. If the embedding of a digraph is given, upward planarity can be tested in polynomial time [2].

k-slope drawings. A *k-slope drawing* of a graph G is a straight-line drawing of G where every edge is drawn with one of at most k different slopes; see Fig. 1a–b. The *(planar) slope number* of G is the smallest k such that G admits a planar k-slope drawing. If only planar drawings are allowed, this is called the *planar slope number* of G. Both numbers have been studied extensively for a variety of classes [4, 11, 13, 15, 16, 20, 21, 25, 27–29, 32, 34]. Determining the planar slope number of a graph is hard in the existential theory of the reals [19].

Di Battista and Tamassia [10] have shown that if a digraph is upward planar, then it even admits an upward planar straight-line drawing. We may therefore ask how many slopes are necessary for a digraph G to admit an upward planar drawing. The answer to this question defines the *upward planar slope number* of G. To the best of our knowledge, this number has so far only been studied by Czyzowicz et al. [6, 7] for lattices and, allowing one bend per edge, by Di Giacomo et al. [12] for series-parallel digraphs and by Bekos et al. [1] for st-graphs.

Most research on slope numbers is concerned with the minimum number of slopes required to draw any graph of a fixed class. We can consider the problem also from a different perspective, namely, given only k slopes, decide whether a given digraph admits an *upward*

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

planar k-slope drawing. For $k = 2$, this question has been investigated by Klawitter and Mchedlidze [23] who show that deciding whether a given upward plane digraph admits an upward planar 2-slope drawing can be done in linear time. For the variable embedding scenario, they give a linear-time algorithm for single-source digraphs, a quartic-time algorithm for series-parallel digraphs, and a fixed-parameter tractable algorithm for general digraphs. Here, we investigate the next natural case $k = 3$. Note that a 2-slope drawing can be stretched in the direction of one slope without affecting the length of edges drawn with the other slope. The fact that this does not hold for three (or more) slopes introduces interesting new geometric aspects for 3-slope drawings.

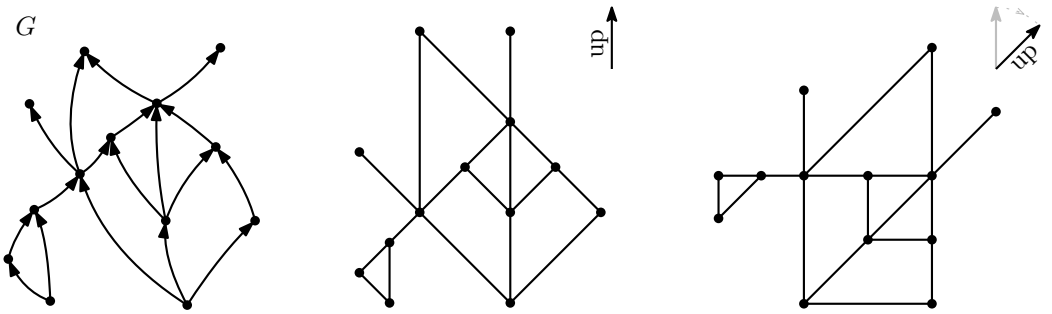


Figure 1 (a) A digraph G with **(b)** upward planar 3-slope drawing; **(c)** drawing rotated by 45°.

Note that the maximum indegree and outdegree of a digraph G are natural lower bounds on its upward planar slope number. Since we have $k = 3$, we assume that our digraphs have maximum indegree and outdegree at most three. Further note that an upward planar 3-slope drawing of a digraph G on any three slopes can be affinely transformed into one with slopes 45°, 90°, and 135°. Hence, we restrict considerations to this slope set. For illustrative purposes however, we rotate drawings by 45° clockwise and thus use the slope set $\{\uparrow, \nearrow, \rightarrow\}$; see Fig. 1c. A *3-slope assignment* of a digraph G assigns each edge of G one of the slopes in $\{\uparrow, \nearrow, \rightarrow\}$. If G is upward plane, we say a 3-slope assignment of G is *consistent* if the assignment complies with the cyclic edge order around each vertex; for example, if a vertex has three incoming edges, then they need to be assigned the slopes $\rightarrow$, $\nearrow$, and $\uparrow$ in ccw order. Clearly, if an upward plane embedding does not admit a consistent 3-slope assignment, then it also does not admit an upward plane 3-slope drawing.

Contribution. We initiate research on upward planar 3-slope drawings by examining the complexity of the decision problem. Some of our results are generalizable to $k \geq 3$ slopes. We classify whether an embedded tree T admits an upward planar 3-slope drawing. In the affirmative, we can construct such a drawing. Otherwise, we can change the embedding of T such that it admits a desired drawing. We further show that it is NP-hard to decide whether a given upward outerplanar[1] digraph admits an upward planar 3-slope drawing.

For statements marked with "⋆", a proof is available on arXiv [24].

[1] An *upward outerplanar* digraph admits an upward planar drawing with each vertex on the outer face.

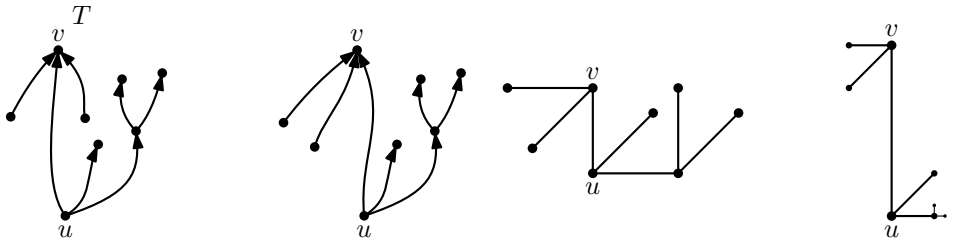


Figure 2 (a) The tree T does not admit a consistent 3-slope assignment, **(b)** unless we change the embedding; **(c)** an upward planar 3-slope drawing of T with exponentially shrinking edge lengths.

2 Trees

We start with the class of directed trees[2] with indegree and outdegree at most three. While every bimodal embedding of a tree induces an upward planar embedding, it does not necessarily admit an upward planar 3-slope drawing; see tree T in Fig. 2a. There the edge uv of T needs to get slope $\uparrow$ by u but slope $\nearrow$ by v and thus obstructs a consistent 3-slope assignment. However, we can always change the embedding of T such that it admits a consistent 3-slope assignment and drawing; see Fig. 2b. We show that both testing whether an embedding is suitable and finding a suitable embedding can be accomplished in linear time.

▶ **Theorem 2.1** ($\star$). *Under a fixed bimodal embedding, a directed tree T admits an upward planar 3-slope drawing if and only if it admits a consistent 3-slope assignment. Moreover, this can be tested in linear time and, in the affirmative, a drawing can be constructed.*

Proof sketch. While the "only-if"-direction is clear, the converse can be shown by drawing T with edge lengths exponentially shrinking as the distance from a start edge increases; see Fig. 2c. A consistent 3-slope assignment is either directly obstructed by an edge as in Fig. 2a or can be computed straightforwardly. ◀

▶ **Theorem 2.2** ($\star$). *Given a directed tree T with indegree and outdegree at most three, we can construct an upward planar 3-slope drawing of T in linear time.*

Proof sketch. To find a suitable embedding, pick a start vertex, order its incident edges bimodally, and continue vertex by vertex to construct a consistent 3-slope assignment. ◀

We conjecture that a similar approach also works for cactus graphs, though constructing an actual drawing is geometrically more involved. We will investigate this further.

3 Outerplanar Graphs

In this section, we show that already for upward outerplanar digraphs, it is NP-hard to decide whether a digraph admits an upward planar 3-slope drawing. It remains open whether the problem is also NP-complete since containment in NP is not immediately clear. More precisely, if the problem was in NP, there would be small proof certificates for yes-instances that a verifier could use to decide the problem in polynomial time. Typically a combinatorial characterization or a drawing of the input graph could act as such a certificate. However in our

[2] A *directed tree* (or *polytree*) is a digraph whose underlying undirected graph is a tree.

case, we do not know whether there are graphs that require irrational (or super-polynomial precise) coordinates and if so, how to treat them implicitly. Yet NP-hardness holds true, even if an upward outerplanar embedding is given. Remember that for an arbitrary number of slopes, upward planarity for outerplanar digraphs can be decided in polynomial time [33] – if an embedding is given, even in linear time. We show NP-hardness by reduction from Planar Monotone 3-SAT [8], an NP-complete version of 3-SAT, where the three literals of each clause are all either negated or unnegated – from now on called *negative* and *positive* clauses, respectively. Moreover, the incidence graph[3] is planar and has a planar drawing where the vertices are rectangles, the edges are vertical straight-line segments, the variables are arranged on a horizontal line, the positive clauses are above, and the negative clauses are below this line. For an example, see Fig. 5a. For a given formula F and a rectangular drawing of its incidence graph, we construct a corresponding upward outerplanar digraph G_F, which can only be drawn upward planar with 3 slopes if F is satisfiable. Our construction follows ideas of Nöllenburg [30] and Kraus [26] and utilizes the following observations.

Up to scaling and mirroring diagonally, $G_\square$ in Fig. 3a admits an upward planar 3-slope drawing only as an outerplanar square as in Fig. 3b. We can attach multiple squares (and triangles) to each other as in Fig. 3c–d. The drawing of such a bigger digraph is unique up to scaling and mirroring diagonally. If the squares form a tree, the drawing is outerplanar. We refer to these squares as *unit squares*, since, once set, the side lengths for all attached squares are the same. To allow a certain small degree of freedom, we exploit the following.

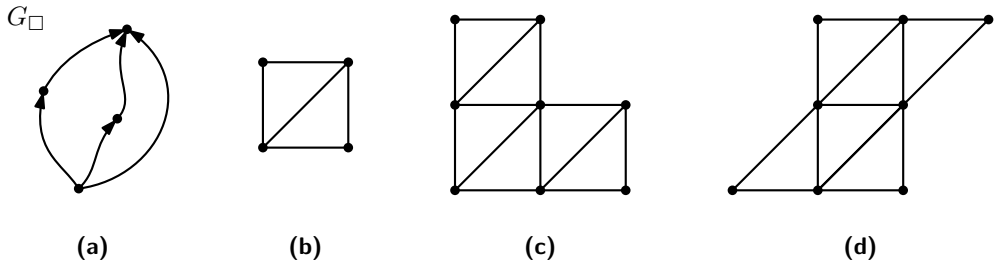


Figure 3 (a) The digraph $G_\square$ admits only an upward 3-slope drawing as square **(b)**; **(c, d)** by combining copies of $G_\square$ and triangles we can build larger rigid structures.

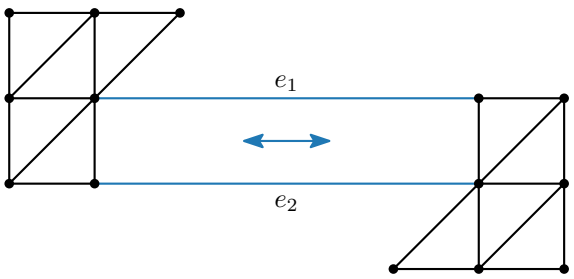
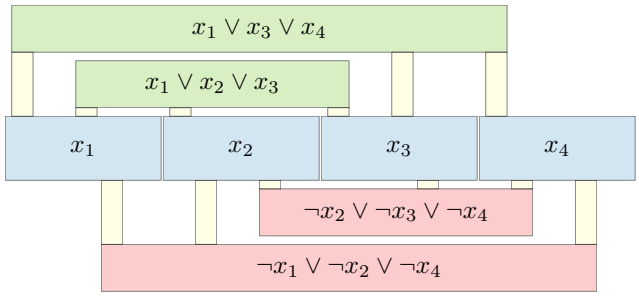


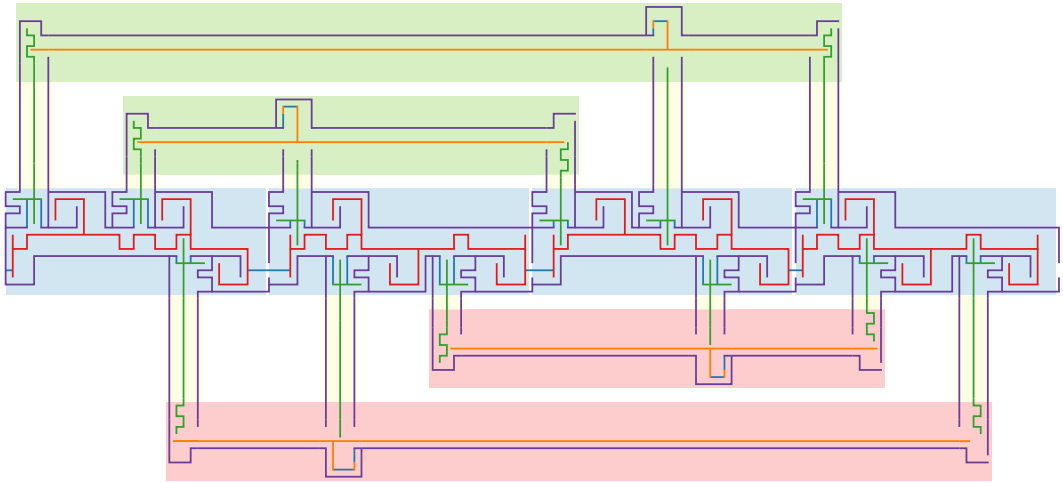
Figure 4 Upward planar 3-slope drawing of the digraph $G_\leftrightarrow$.

▶ **Lemma 3.1** ($\star$). *In any upward planar 3-slope drawing of the digraph $G_\leftrightarrow$ (see Fig. 4), ...*

[3] In the incidence graph of a SAT formula, there is a vertex for each variable and each clause, and for each occurrence of a variable in a clause, there is an edge between the corresponding vertices.



(a) Rectangular drawing of the incidence graph of a Planar Monotone 3-SAT formula F.



(b) Outerplanar drawing of the digraph G_F obtained from (a). Chains of unit squares are drawn as straight line segments. The variable/clause/edge gadgets occupy the areas of their corresponding rectangles. Here, x_1 and x_4 are set to false (brush on the left), while x_2 and x_3 are set to true (brush on the right)

Figure 5 Schematic example for our NP-hardness reduction from Planar Monotone 3-SAT.

- *the edges e_1 and e_2 are parallel and have the same arbitrary length $\ell > 0$,*
- *all other edges are oriented as in Fig. 4 up to mirroring along a diagonal axis, and*
- *all other vertical and horizontal edges have the same lengths, as well as all diagonal edges.*

With this construction kit of useful (sub)graphs in hand, we build a bigger graph whose upward planar drawings represent the satisfying truth assignments for F. The **high-level construction** is depicted in Fig. 5b. We construct, for each variable x_i, an individual digraph – the *variable gadget for x_i*. Similarly, for each clause c_j, there is an individual digraph – the *clause gadget for c_j*. All gadgets mainly consist of chains of $G_{\square}$s. For a drawing, this enforces a rigid frame structure built from unit squares. We glue all variable gadgets together in a row and connect variable and clause gadgets by *edge gadgets* such that the composite graph remains upward outerplanar (see Fig. 5b) and all $G_{\square}$s are drawn as unit squares.

A **variable gadget** is depicted in Fig. 6. Its base structure is the (violet) frame composed of chains of unit squares. The core element is the (red) central chain of unit squares (with a few side-arms), which has one degree of flexibility, namely, moving as a whole to the left or to the right without leaving the frame structure of the gadget. It looks and behaves a bit like a pipe cleaning brush that is stuck inside the frame but can be moved a bit back and forth. Hence, we also call it a *brush*. It is connected via a $G_{\leftrightarrow}$ to the brush of the previous variable

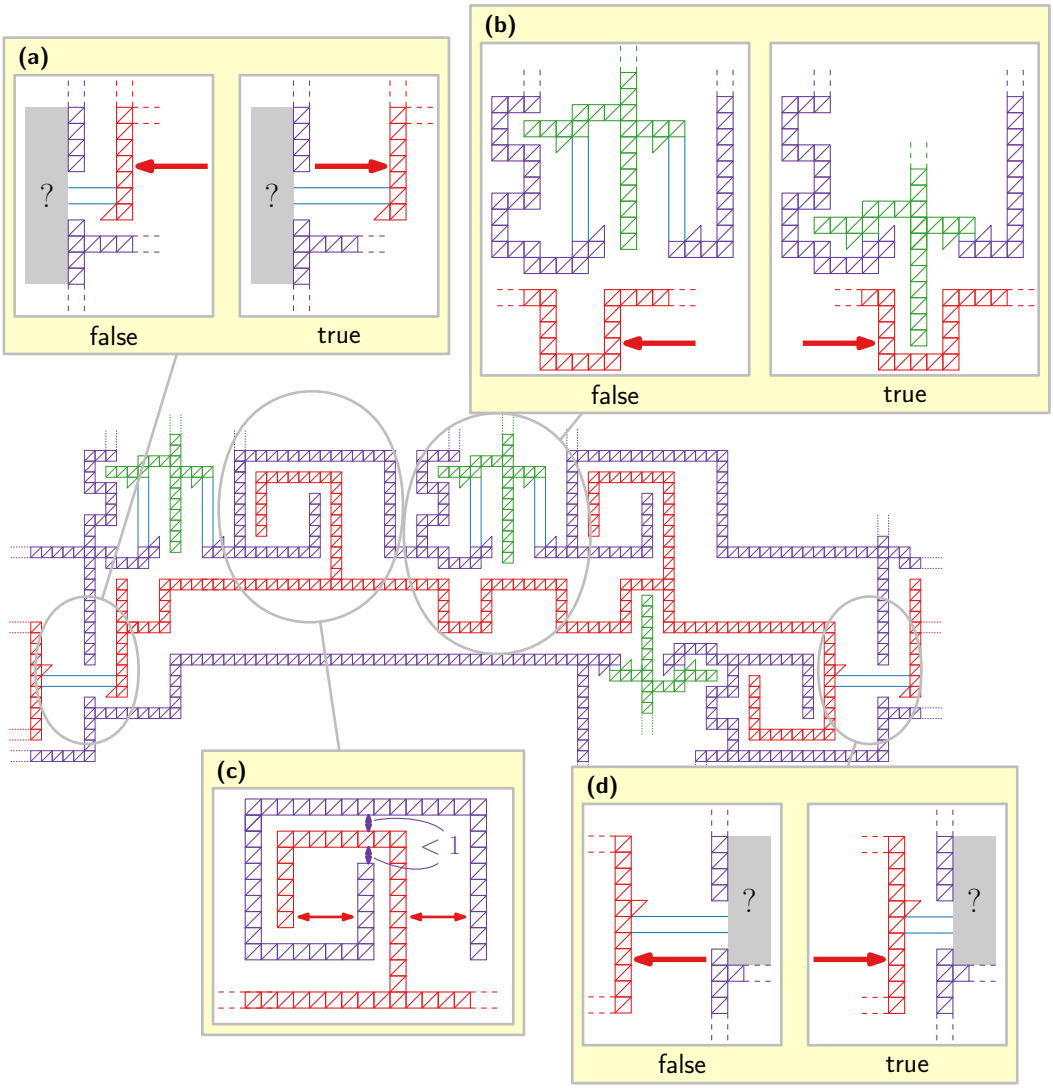


Figure 6 A variable gadget, contained in two positive and one negative clauses, being set to false.

gadget (see Fig. 6a/d) and the first brush is connected to the frame via a $G_{\leftrightarrow}$ (see the left of Fig. 5b). This allows only a horizontal shift of the brushes, but no vertical movement relative to its anchor point at the frame structure. Note that the horizontal position in any variable gadget is independent of those in all other gadgets. If the brush is positioned close to the very left (right), the corresponding variable is set to false (true).

For each occurrence of a variable in a positive clause, we have a construction as depicted in Fig. 6b. There, a long chain of (green) $G_{\square}$s – from now on called *bolt* – is attached to the frame structure via two $G_{\leftrightarrow}$s, which allow only a vertical, but no horizontal shift. The bolt has on its left side an arm, which can only be placed in one of two pockets of the frame. It can always be placed in the upper pocket, which pushes the bolt outwards with respect to the variable gadget (into an edge and then a clause gadget). It can only be placed in the lower pocket if the brush is shifted close to the very right (i.e. set to true) – then the bolt can "fall" into a cove of the brush. For each occurrence of a variable in a negative clause, we have the same construction, but upside-down, such that the bolt can be pulled into the

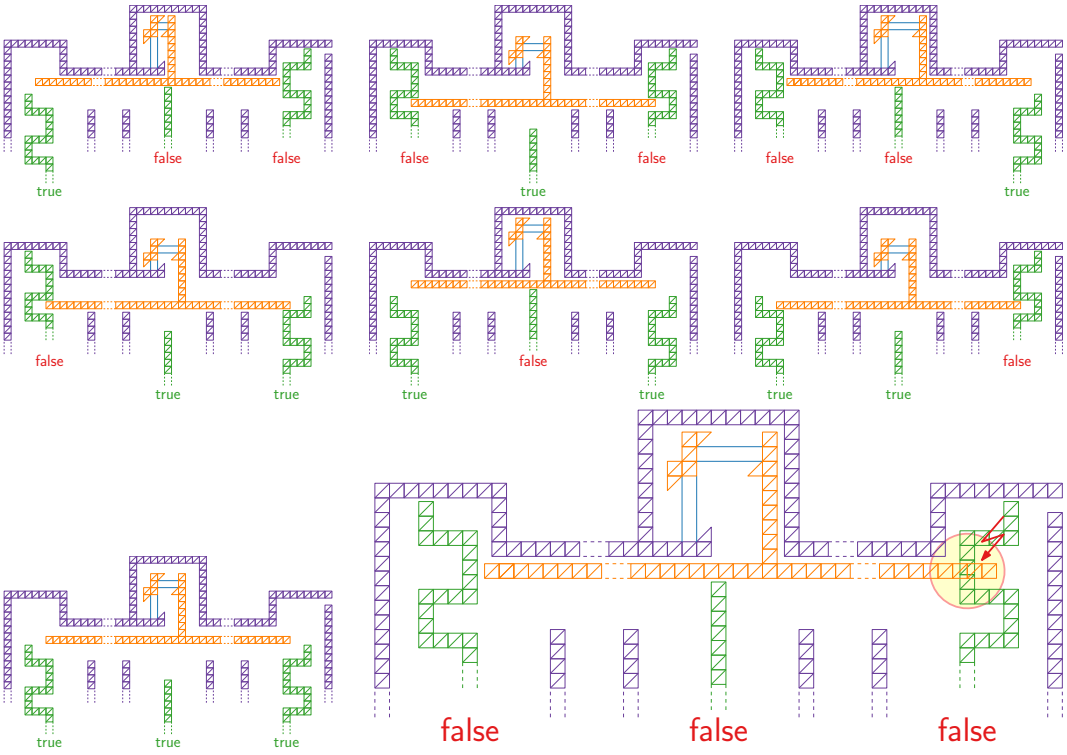


Figure 7 Positive clause gadget (negative clause gadget is mirrored vertically) in 8 configurations.

variable gadget only if the brush is shifted close to the very left (i.e. set to false).

Note that, to maintain outerplanarity of the whole construction, the frame structure is not contiguous, but connected by $G_{\leftrightarrow}$s and the arms of the bolts. Hence, the frame structure decomposes into many components that have fixed relative horizontal positions and their unit squares have the same side lengths. However, the components can shift up and down relative to each other. To keep this vertical shift small enough not to affect the correct functioning of our reduction, we use, for each such component, the construction depicted in Fig. 6c. The chain of brushes has no flexibility in vertical direction and serves as a base ground for an "anchor" of the frame structure. The frame structure can move less than one unit up or down unless it violates planarity. If the frame structure would be shifted up enough to be completely above the brush, it would get in conflict with the adjacent bolt.

An **edge gadget** consists of only three straights chains – two are frame segments, one is a bolt in the middle. Their purpose is to synchronize the distance of the clause gadgets to the variable gadgets and to provide these chains of unit squares for the clause gadgets. Several edge gadgets are depicted on yellow background color in Fig. 5b.

A **clause gadget** for a positive clause is depicted in Fig. 7. Within a frame, which is connected at six points to the frames of three edge gadgets, there is a horizontal (orange) bar, which is attached via two $G_{\leftrightarrow}$s to the frame – one $G_{\leftrightarrow}$ allows a horizontal, the other allows a vertical shift. It resembles a crane that can move up and extend its arm, while it holds the horizontal bar on a vertical (orange) rope. The three bolts from the corresponding variable gadgets reach into the clause gadget. The lengths of these bolts is chosen such that, if they are pushed out of their variable gadget and into the clause gadget, they only slightly fit inside the gadget. Depending on whether each of the bolts is pushed into the clause gadget or pulled out of it, we have eight possible configurations (with sufficiently small

vertical slack). They represent the eight possible truth assignments to a clause. In Fig. 7, we illustrate that in each configuration, we can accommodate the horizontal bar in an upward planar 3-slope drawing of the clause gadget – except for the case when all three bolts push into the clause gadget, which represents the truth assignment false to all contained variables. A negative clause gadget uses the same construction, but mirrored vertically. There, three bolts pushing into the clause gadget means the contained variables are all set to true.

We conclude with our main theorem. In the future, we plan to generalize Theorem 3.2 to $k \geq 3$ slopes. In the fixed embedding setting, the main idea is to simply add $2(k-3)$ dummy leaves to each vertex that automatically occupy the additional slopes.

▶ **Theorem 3.2** ($\star$). *Deciding whether a directed outerplanar graph admits an upward planar 3-slope drawing is NP-hard, even when an upward outerplanar embedding is given.*

Acknowledgments. We would like to thank the reviewers for their helpful comments.

References

1 Michael A. Bekos, Emilio Di Giacomo, Walter Didimo, Giuseppe Liotta, and Fabrizio Montecchiani. Universal Slope Sets for Upward Planar Drawings. In Therese Biedl and Andreas Kerren, editors, *Graph Drawing and Network Visualization*, pages 77–91. Springer International Publishing, 2018. doi:10.1007/978-3-030-04414-5_6.

2 Paola Bertolazzi, Giuseppe Di Battista, Giuseppe Liotta, and Carlo Mannino. Upward drawings of triconnected digraphs. *Algorithmica*, 12(6):476–497, 1994. doi:10.1007/BF01188716.

3 Paola Bertolazzi, Giuseppe Di Battista, Carlo Mannino, and Roberto Tamassia. Optimal Upward Planarity Testing of Single-Source Digraphs. *SIAM Journal on Computing*, 27(1):132–169, 1998. doi:10.1137/S0097539794279626.

4 Guido Brückner, Nadine Davina Krisam, and Tamara Mchedlidze. Level-planar drawings with few slopes. In Daniel Archambault and Csaba D. Tóth, editors, *Graph Drawing and Network Visualization*, pages 559–572, 2019. doi:10.1007/978-3-030-35802-0_42.

5 Hubert Chan. A Parameterized Algorithm for Upward Planarity Testing. In Susanne Albers and Tomasz Radzik, editors, *Algorithms – ESA 2004*, pages 157–168. Springer, 2004. doi:10.1007/978-3-540-30140-0_16.

6 Jurek Czyzowicz. Lattice diagrams with few slopes. *Journal of Combinatorial Theory, Series A*, 56(1):96–108, 1991. doi:10.1016/0097-3165(91)90025-C.

7 Jurek Czyzowicz, Andrzej Pelc, and Ivan Rival. Drawing orders with few slopes. *Discrete Mathematics*, 82(3):233–250, 1990. doi:10.1016/0012-365X(90)90201-R.

8 Mark de Berg and Amirali Khosravi. Optimal binary space partitions for segments in the plane. *International Journal of Computational Geometry & Applications*, 22(3):187–206, 2012. doi:10.1142/s0218195912500045.

9 Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice-Hall, 1999.

10 Giuseppe Di Battista and Roberto Tamassia. Algorithms for plane representations of acyclic digraphs. *Theoretical Computer Science*, 61(2):175–198, 1988. doi:10.1016/0304-3975(88)90123-5.

11 Emilio Di Giacomo, Giuseppe Liotta, and Fabrizio Montecchiani. Drawing Outer 1-planar Graphs with Few Slopes. In Christian Duncan and Antonios Symvonis, editors, *Graph Drawing*, pages 174–185. Springer, 2014. doi:10.1007/978-3-662-45803-7_15.

12 Emilio Di Giacomo, Giuseppe Liotta, and Fabrizio Montecchiani. 1-Bend Upward Planar Drawings of SP-Digraphs. In Yifan Hu and Martin Nöllenburg, editors, *Graph Drawing*

and Network Visualization, pages 123–130. Springer International Publishing, 2016. doi:10.1007/978-3-319-50106-2_10.

13 Emilio Di Giacomo, Giuseppe Liotta, and Fabrizio Montecchiani. Drawing subcubic planar graphs with four slopes and optimal angular resolution. *Theoretical Computer Science*, 714:51–73, 2018. doi:10.1016/j.tcs.2017.12.004.

14 Walter Didimo, Francesco Giordano, and Giuseppe Liotta. Upward Spirality and Upward Planarity Testing. *SIAM Journal on Discrete Mathematics*, 23(4):1842–1899, 2010. doi:10.1137/070696854.

15 Vida Dujmović, David Eppstein, Matthew Suderman, and David R. Wood. Drawings of planar graphs with few slopes and segments. *Computational Geometry*, 38(3):194–212, 2007. doi:10.1016/j.comgeo.2006.09.002.

16 Vida Dujmović, Matthew Suderman, and David R. Wood. Graph drawings with few slopes. *Computational Geometry*, 38(3):181–193, 2007. doi:10.1016/j.comgeo.2006.08.002.

17 Ashim Garg and Roberto Tamassia. On the Computational Complexity of Upward and Rectilinear Planarity Testing. *SIAM Journal on Computing*, 31(2):601–625, 2001. doi:10.1137/S0097539794277123.

18 Patrick Healy and Karol Lynch. Two fixed-parameter tractable algorithms for testing upward planarity. *International Journal of Foundations of Computer Science*, 17(05):1095–1114, 2006. doi:10.1142/S0129054106004285.

19 Udo Hoffmann. On the Complexity of the Planar Slope Number Problem. *Journal of Graph Algorithms and Applications*, 21(2):183–193, 2017. doi:10.7155/jgaa.00411.

20 Vít Jelínek, Eva Jelínková, Jan Kratochvíl, Bernard Lidický, Marek Tesař, and Tomáš Vyskočil. The Planar Slope Number of Planar Partial 3-Trees of Bounded Degree. *Graphs and Combinatorics*, 29(4):981–1005, 2013. doi:10.1007/s00373-012-1157-z.

21 Balázs Keszegh, János Pach, and Dömötör Pálvölgyi. Drawing Planar Graphs of Bounded Degree with Few Slopes. *SIAM Journal on Discrete Mathematics*, 27(2):1171–1183, 2013. doi:10.1137/100815001.

22 Philipp Kindermann, Wouter Meulemans, and André Schulz. Experimental analysis of the accessibility of drawings with few segments. *Journal of Graph Algorithms and Applications*, 22(3):501–518, 2018. doi:10.7155/jgaa.00474.

23 Jonathan Klawitter and Tamara Mchedlidze. Upward planar drawings with two slopes, 2021. In preparation.

24 Jonathan Klawitter and Johannes Zink. Upward planar drawings with three slopes, 2021. arXiv:2103.06801.

25 Kolja Knauer, Piotr Micek, and Bartosz Walczak. Outerplanar graph drawings with few slopes. *Computational Geometry*, 47(5):614–624, 2014. doi:doi.org/10.1016/j.comgeo.2014.01.003.

26 Rebecca Kraus. Level-außenplanare Zeichnungen mit wenigen Steigungen, 2020. Bachelor Thesis, University of Würzburg.

27 William Lenhart, Giuseppe Liotta, Debajyoti Mondal, and Rahnuma Islam Nishat. Planar and Plane Slope Number of Partial 2-Trees. In Stephen Wismath and Alexander Wolff, editors, *Graph Drawing*, pages 412–423. Springer, 2013. doi:10.1007/978-3-319-03841-4_36.

28 Padmini Mukkamala and Dömötör Pálvölgyi. Drawing Cubic Graphs with the Four Basic Slopes. In Marc van Kreveld and Bettina Speckmann, editors, *Graph Drawing*, pages 254–265. Springer, 2012. doi:10.1007/978-3-642-25878-7_25.

29 Padmini Mukkamala and Mario Szegedy. Geometric representation of cubic graphs with four directions. *Computational Geometry*, 42(9):842–851, 2009. doi:10.1016/j.comgeo.2009.01.005.

30 Martin Nöllenburg. Automated Drawing of Metro Maps, 2010. Diploma Thesis, University of Karlsruhe (TH).

31 Martin Nöllenburg and Alexander Wolff. Drawing and labeling high-quality metro maps by mixed-integer programming. *IEEE Transactions on Visualization and Computer Graphics*, 17(5):626–641, 2011. doi:10.1109/TVCG.2010.81.

32 János Pach and Dömötör Pálvölgyi. Bounded-degree graphs can have arbitrarily large slope numbers. *Electronic Journal of Combinatorics*, 13(1):N1, 2006.

33 Achilleas Papakostas. Upward planarity testing of outerplanar dags. In Roberto Tamassia and Ioannis G. Tollis, editors, *Graph Drawing*, pages 298–306. Springer, 1995. doi:10.1007/3-540-58950-3_385.

34 G. A. Wade and J.-H. Chu. Drawability of Complete Graphs Using a Minimal Slope Set. *The Computer Journal*, 37(2):139–142, 1994. doi:10.1093/comjnl/37.2.139.

Decomposing Polygons into Fat Components

Maike Buchin[1] **and Leonie Selbach**[1]

1 **Ruhr University Bochum**
maike.buchin@rub.de, leonie.selbach@rub.de

Abstract

We study the problem of decomposing (i.e. partitioning or covering) polygons into components that are α-fat, which means that the aspect ratio of each subpolygon is at most α. We consider decompositions without Steiner points. We present a polynomial-time algorithm for simple polygons that finds the minimum α such that an α-fat partition exists. Furthermore, we show that finding an α-fat partition or covering with minimum cardinality is NP-hard for polygons with holes.

Related Version A full version of this paper is available at `https://arxiv.org/abs/2103.08995` [1]

1 Introduction

A decomposition of a polygon P is a set of subpolygons whose union is exactly P. If the subpolygons are not allowed to overlap, the set is a *partition* of P. Otherwise, we call the set a *covering*. Here, we consider decompositions without Steiner points. Thus, a polygon is decomposed by adding diagonals between its vertices. Polygon decomposition problems arise in many theoretical and practical applications and can be categorized with regard to the type of subpolygon that is used [6]. We consider decompositions into *fat components*. Our research is motivated by a bioinformatical application: The fragmentation of tissue samples can be modeled as a constrained shape decomposition problem in which specifically round or fat components are desired [7].

A polygon P is called α-*fat* if its aspect ratio (AR) is at most α. There are different definitions for the aspect ratio and in this paper we consider the following two (see Fig. 1):

square-fatness: AR_{square} = ratio between the side length of the smallest axis-parallel square containing P and side length of the largest axis-parallel square contained in P [5].

disk-fatness: AR_{disk} = ratio between the diameter of the smallest circle enclosing P (minimum circumscribed circle or MCC) and the diameter of the largest circle enclosed in P (maximum inscribed circle or MIC) [3].

A polygon P is called α-*small* if the side length of the enclosing square or respectively the diameter of the enclosing circle is at most α. The *minimum* α-*fat partition* (or *covering*) problem is finding a partition (or covering) with minimum cardinality such that every subpolygon is α-fat. We have analog problems with α-smallness instead of α-fatness.

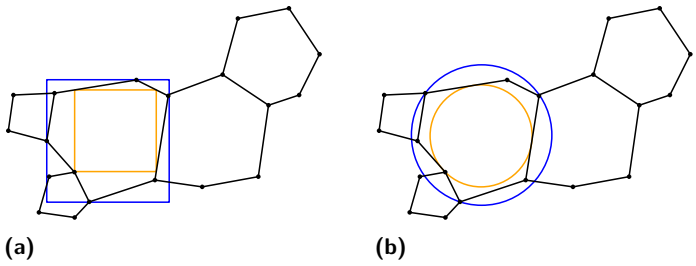


Figure 1 Partition that is 1.5-fat with square-fatness (a) and 1.4-fat with disk-fatness (b).

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Worman showed that the minimum α-small partition problem as well as the covering problem are NP-hard for polygons with holes [8]. Square-smallness was used for the reduction, but with some adjustments the construction holds for disk-smallness as well [1]. In the following results disk-smallness and -fatness was used. Damian and Pemmaraju showed that the minimum α-small partition problem is polynomial-time solvable for simple polygons and that a faster 2-approximation algorithm exists [3]. Additionally, the authors presented an approximation algorithm for convex polygons. Damian proved that the minimum α-fat partition problem can be solved in polynomial time for simple polygons and conjectured that this problem is NP-hard for polygons with holes [2]. The *min-fat partition* problem is finding the smallest α for which an α-fat partition exists. Solving the min-fat partition problem was stated as an open problem by Damian [4].

This paper includes two main results. In section 2 consider the min-fat partition problem using disk-fatness and present a polynomial-time algorithms for simple polygons. In section 3 we consider the minimum α-fat partition and covering problem and confirm the conjecture that these problems are NP-hard for polygons with holes. This result is true for both square- and disk-fatness. Because of the line constraints, we present only the reduction for square-fatness. All missing proofs and results can be found in the full version of this paper [1].

2 Min-fat partition problem for simple polygons

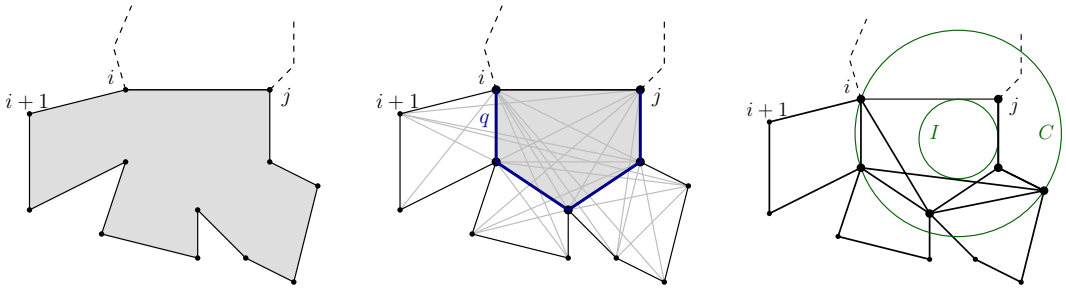
We extend the method of Damian [2] to solve the min-fat partition problem for simple polygons while using the definition of disk-fatness. Our goal is to find a partition such that the largest aspect ratio (AR) of any subpolygon is minimized. Note that the value of the largest AR in this optimal partition equals the desired α in the min-fat partition problem.

Let P be a simple polygon, with vertices labeled from 1 to n counterclockwise. A diagonal (i, j) is a line segment that connects two vertices i and j and does not intersect the outside of P. Let $G(P)$ be the visibility graph of P consisting of the n vertices of P and m diagonals. We define S as the set consisting of all vertices and edges of $G(P)$. For each diagonal (i, j) with $i < j$, let $P_{i,j}$ be the subpolygon with vertices $\{i, i+1, \ldots, j\}$ (see Fig. 2a). To solve the min-fat partition problem, we compute an optimal partition $Z_{i,j}$ for each $P_{i,j}$. This can be done iteratively. Let Q be the polygon in $Z_{i,j}$ adjacent to (i, j). Note that the vertices of Q induce a path q from i to j in the visibility graph (see Fig. 2b) and we have $Z_{i,j} = \bigcup_{(k,l)\in q} Z_{k,l} \cup Q$. The idea is to find an optimal partition by computing the optimal path q. We define edge weights $w(i, j)$ as the value of am optimal partition $Z_{i,j}$ of $P_{i,j}$:

$$w(i,j) = \max_{P' \in Z_{i,j}} AR(P') = \max\{\max_{(k,l)\in q} w(k,l), AR(Q)\}$$

for $AR(Q) = d(MCC)/d(MIC)$ being the ratio between the diameters $d(\cdot)$ of the minimum circumscribed circle MCC and maximum inscribed circle MIC of Q. If j is equal to $i + 1$, the partition $Z_{i,i+1}$ is empty and we set $w(i, i + 1) = 0$. Otherwise, $w(i, j)$ equals the value of the largest AR in an min-fat partition of $P_{i,j}$.

However, the computation of $w(i, j)$ presents the following problem: Finding the path q and its correct edges requires knowledge about the resulting polygon Q and its aspect ratio, which is obviously not available beforehand. Therefore, we compute paths on different reduced graphs that ensure that the aspect ratio of each possible polygon is below a certain value and then choose the best one. For this, we consider all pairs of circles (C, I) such that the following properties hold: (i, j) lies completely inside of C and outside of I, I is tangent to 3 elements in S, and C either touches 3 vertices of P or its diameter connects 2



(a) $P_{i,j}$. **(b)** Polygon Q. **(c)** Graph $G_{i,j}^{(C,I)}$.

Figure 2 In (a): Subpolygon $P_{i,j}$. In (b): Polygon Q (gray) in an optimal partition of $P_{i,j}$ induced by a path q (blue edges) in the visibility graph. In (c): Reduced visibility graph $G_{i,j}^{(C,I)}$ (fat edges) for a pair of circles (C, I).

vertices. Note that for any subpolygon the pair (MCC,MIC) fulfills these properties. For each (C, I), let $G_{i,j}^{(C,I)}$ be the subgraph of $G(P)$ consisting of edges that lie inside of $P_{i,j}$ and C and outside of I (see Fig. 2c).

We can compute the weights $w(i, j)$ by using dynamic programming with increasing values of $j - i$. For each pair of circles (C, I), we compute a corresponding weight $W(C, I)$ and use those values to determine an edge weight $w'(i, j)$ as follows:

$$w'(i,j) = \min W(C,I) = \min \min_{q \in G_{i,j}^{(C,I)}} \max\{ \max_{(k,l)\in q} w'(i,j), d(C)/d(I)\}.$$

The weights of all edges except for (i, j) have already been computed. We search in $G_{i,j}^{(C,I)}$ for the path q such that the value $\max\{\max_{(k,l)\in q} w'(k, l), d(C)/d(I)\}$ is minimized. We denote this optimal value as $W(C, I)$. Over all possible combinations of circles, we search for the pair (C, I) with minimum $W(C, I)$ and set $w'(i, j) = W(C, I)$. We can show by induction that $w'(i, j)$ is actually equal to the largest aspect ratio in the corresponding partition and that this partition is indeed optimal and thus $w'(i, j) = w(i, j)$.

▶ **Lemma 2.1.** *For an edge (i, j) in the visibility graph $G(P)$ with $j \neq i+1$, let $w'(i, j)$ be the computed weight and $Z_{i,j}$ the corresponding partition. Then, $w'(i, j) = \max_{P' \in Z_{i,j}} AR(P')$.*

▶ **Lemma 2.2.** *The computed partition $Z_{i,j}$ is an optimal partition of $P_{i,j}$, meaning that the largest aspect ratio of any subpolygon is minimized.*

▶ **Theorem 2.3.** *For a simple polygon P, the min-fat partition problem using disk-fatness can be solved in time $\mathcal{O}(n^3 m^5 \log n)$ with n being the number of vertices of P and m being the number of edges in the visibility graph $G(P)$.*

Proof. First, we have to compute the visibility graph of P which takes $\mathcal{O}(n + m)$ time. For every edge (i, j) in the visibility graph, we determine an optimal partition $Z_{i,j}$ by computing the optimal weight $w(i, j)$. For each (i, j), we consider pairs of circles (C, I). There are $\mathcal{O}(n^3)$ circles C and $\mathcal{O}(m^3)$ circles I to consider. Computing the optimal path in $G_{i,j}^{(C,I)}$ can be done by an adjusted version of Dijkstra's Algorithm and therefore takes $\mathcal{O}(m \log n)$ time. Thus, the overall runtime of the algorithm is $\mathcal{O}(n^3 m^5 \log n)$. ◀

3 Minimum α-fat decomposition problems for polygons with holes

We show that given a polygon with holes, it is NP-complete to decide whether there exists an α-fat partition or covering with a given number of components. Similarily to Worman [8], we show its NP-hardness by a reduction from *planar 3,4-SAT*. This is the problem of deciding if a boolean formula ϕ is satisfiable under the following 3 restrictions: In conjunctive normal form ϕ has exactly 3 literals per clause, each literal appears in at most 4 clauses, and the graph $G(\phi) = (V, E)$ with $V = U \cup C$ and $E = \{(u, c) \mid u \in U, c \in C, u \text{ or } \bar{u} \text{ is a literal in } c\}$, where U are the literals and C the clauses, is planar.

We construct a polygon representing the graph $G(\phi)$ that has an α-fat partition of size k if and only if ϕ is satisfiable. We determine the value k during this construction. Our construction uses a fixed value of α and consists of 3 different polygon components: variable, wire and clause polygons. In contrast to the related reduction of Worman, the constructions differ depending on the definition of fatness that is applied. Here, we present the method for square-fatness, the one for disk-fatness can be found in the full version [1].

We set $\alpha = 1.2$. All polygon components have a width of at most 5, thus the side length of the enclosing squares cannot exceed 6. The *variable polygon* is shown in Figure 3a. It has four terminals at which wires can be connected. With the given α, this polygon can be minimally partitioned with 8 subpolygons in two ways (see Fig. 3b). These partitions represent the `True` and `False` assignment that will be carried to the corresponding clauses.

Each wire consists of a set of individual *wire polygons* that are connected with each other (as depicted in Figure 4) to carry the variable assignment to the clause polygon. The wires can be attached at the terminals in two possible orientations (see Fig. 5) depending on whether the variable appears in the clause negated or unnegated. If the wire is connected in the unnegated orientation and the variable is set to `True`, the green polygon in Figure 5a can cover the top part of the wire as well, but this is not the case if the variable is set to `False`. The reverse is true if the wire is connected in the negated orientation. If a wire is partitioned in this way (unnegated position and `True` assignment or negated position and `False` assignment), we say that it carries `True`.

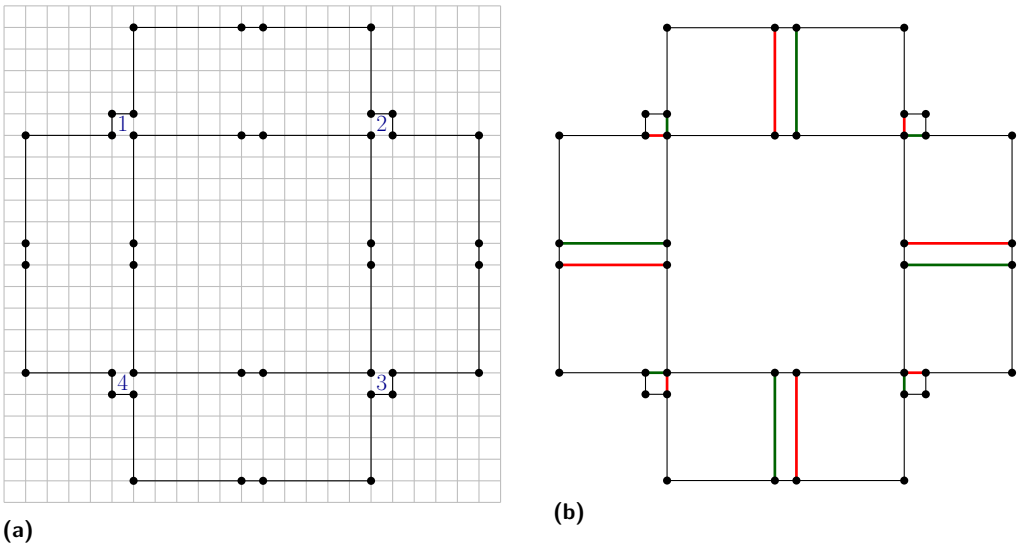
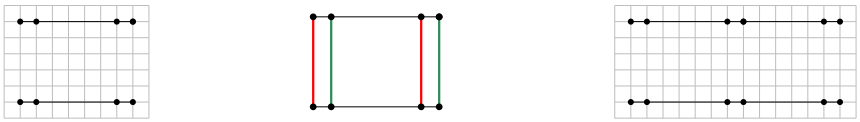
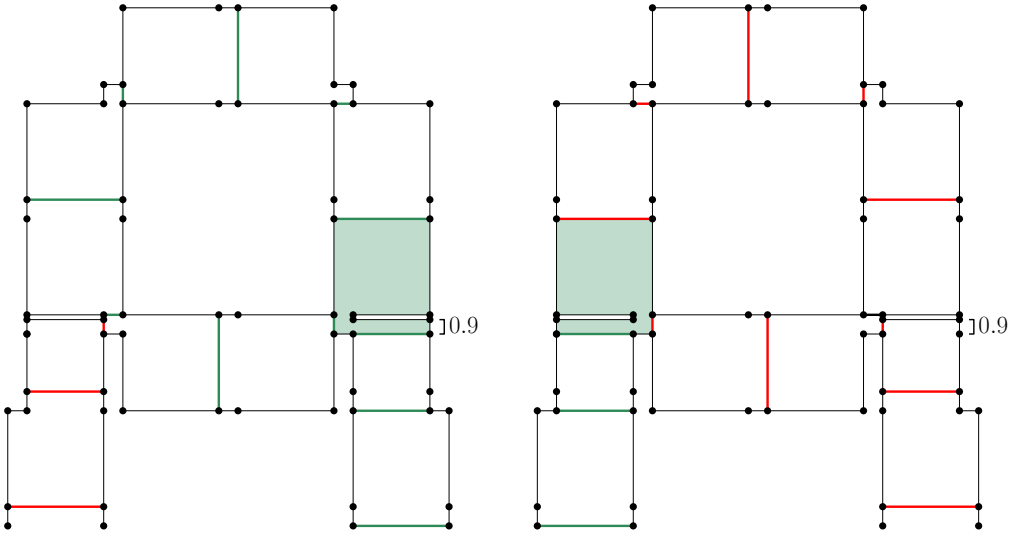


Figure 3 The variable polygon with four terminals indicated by $1, 2, 3, 4$ in blue (a) and its minimal 1.2-fat partitions representing the `True` (green) and `False` (red) assignment (b).



(a) (b) (c)

Figure 4 A single wire polygon (a), its two partitions that represent the `True` (green) or `False` (red) assignment (b), and two connected wire polygons (c).



(a) `True` assignment of variable. **(b)** `False` assignment of variable.

Figure 5 Attaching a wire to the variable in unnegated orientation (on bottom right terminal) and in negated orientation (on bottom left terminal) which switches the `True`/`False` value.

The variable assignment is carried to the *clause polygon* (see Fig. 6). For each of the three variables contained in the clause, this polygon has one terminal where the wires will be attached. Depending on the values these wires carry, a different number of polygons is needed to partition the clause polygon into 1.2-fat components. If some wire carries `True` (see Fig. 7a and 7b), the tip of the connected terminal (gray) is already covered and center part of the clause polygon (dark green) can be covered as well. If more than one wire carries `True` either one of the corresponding polygons (light green) can cover the center. In either case, the partition requires exactly four polygons. If all wires carry `False` (see Fig. 7c), the a part of the center (red area) cannot be covered by any of the bigger polygons (light red) and thus five polygons are needed to partition the clause polygon.

The whole polygon representing $G(\phi)$ is constructed based on a planar orthogonal grid drawing of $G(\phi)$. That is a planar embedding of the graph such that every vertex is located at an integer grid point, the edges are non-overlapping, and every edge is a chain of orthogonal lines that bend at integer grid points. A schematic example for the placement of variable and clause polygons on the vertices of the drawing can be seen in Figure 8. To construct the edges, the wires have to be bend, shifted or offset. This is achieved by the constructions presented in Figure 9. The drawing of $G(\phi)$ is scaled to accommodate the size of the variable and clause polygons as well as the needed adjustments of the wire polygons.

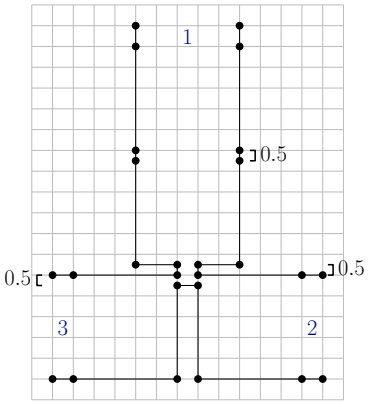
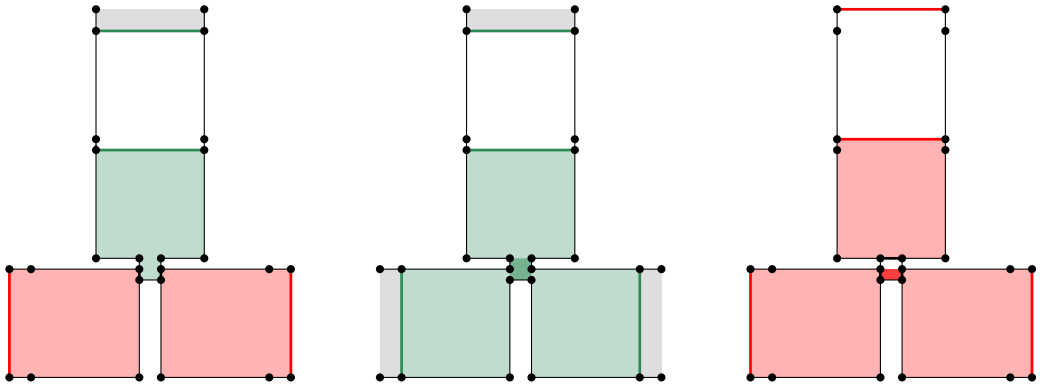


Figure 6 The clause polygon with three terminals indicated by $1, 2, 3$ in blue.



(a) `True`, `False`, `False` **(b)** `True`, `True`, `True` **(c)** `False`, `False`, `False`

Figure 7 Partition of the clause polygon depending on different assignments that are transmitted by the wires (`True` green edges, `False` red edges).

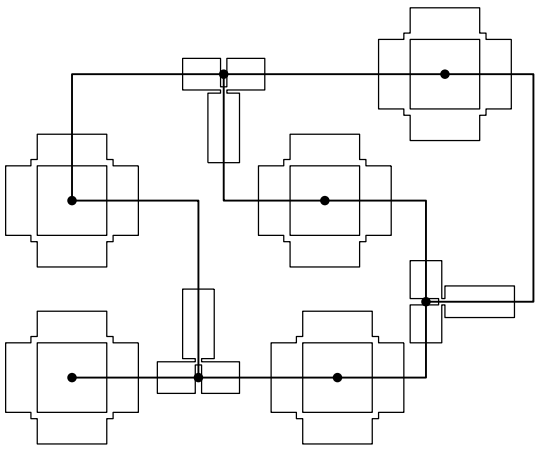


Figure 8 Placement of 5 variable and 3 clause polygons on a planar orthogonal grid drawing.

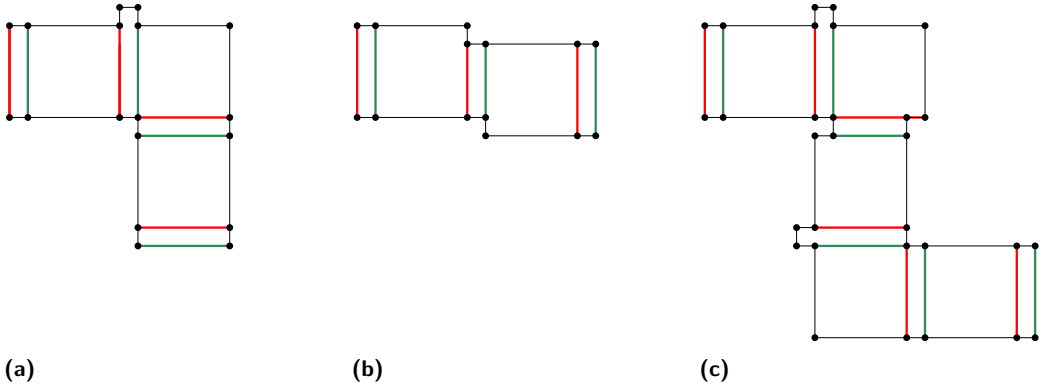


Figure 9 Bending (a), shifting (b) and offsetting (c) a wire that carries `True` (green edges) or `False` (red edges).

As we consider the decision problem, the number k of allowed subpolygons is fixed. We have $k = 8v + 4c + w$ where v is the number of variables, c the number of clauses and w the number of wire polygons needed in the construction. Bending, shifting and offsetting a wire counts as 3, 2, and 5 wire polygons, respectively.

▶ **Theorem 3.1.** *Deciding the α-fat partition problem is NP-complete for polygons with holes if square-fatness is applied.*

Note that we can find a minimum 1.2-fat covering with the same number of components as the minimum partition and that the constructed polygon is orthogonal. Thus, the result remains true for the corresponding covering problem and also for orthogonal polygons with holes. However, the presented construction does not work for disk-fatness: If α is set to the smallest aspect ratio needed such that all components in the construction are still feasible, other subpolygons become feasible, and the transmission of `True`/`False` values would no longer be consistent. We can adjust the construction for disk-fatness, but this adjusted construction in turn does not work for square-fatness anymore [1].

4 Conclusion

We presented a polynomial-time algorithm for the min-fat partition problem for simple polygons. Furthermore, we proved that it is NP-complete to decide the α-fat partition problem and covering problem for polygons with holes. Both results are true for disk-fatness and the latter also holds for square-fatness. It remains open whether the minimum α-fat or min-fat covering problem is solvable for simple polygons. Moreover, there are no results for any related fat decomposition problems that allow Steiner points yet.

References

1 Maike Buchin and Leonie Selbach. Decomposing polygons into fat components, 2021. `arXiv:2103.08995`.
2 Mirela Damian. Exact and approximation algorithms for computing optimal fat decompositions. *Computational Geometry*, 28(1):19–27, 2004.
3 Mirela Damian and Sriram V Pemmaraju. Computing optimal diameter-bounded polygon partitions. *Algorithmica*, 40(1):1–14, 2004.
4 Erik Demaine and Joseph O'Rourke. Open problems from CCCG 2002. In *Proceedings of the 15th Canadian Conference on Computational Geometry*, pages 178–181, 2003.

5 Matthew J Katz. 3-d vertical ray shooting and 2-d point enclosure, range searching, and arc shooting amidst convex fat objects. *Computational Geometry*, 8(6):299–316, 1997.

6 J Mark Keil. Polygon decomposition. *Handbook of computational geometry*, 2:491–518, 2000.

7 Leonie Selbach, Tobias Kowalski, Klaus Gerwert, Maike Buchin, and Axel Mosig. Shape Decomposition Algorithms for Laser Capture Microdissection. In *20th International Workshop on Algorithms in Bioinformatics*, volume 172 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 13:1–13:17, 2020.

8 Chris Worman. Decomposing polygons into diameter bounded components. In *Proceedings of the 15th Canadian Conference on Computational Geometry*, pages 103–106, 2003.

StreamTable: An Area Proportional Visualization for Tables with Flowing Streams*

Jared Espenant and Debajyoti Mondal

Department of Computer Science
University of Saskatchewan, Saskatoon, Saskatchewan, Canada
jae608@usask.ca, d.mondal@usask.ca

Abstract

Let M be an $r \times c$ table with each cell weighted by a nonzero positive number. A StreamTable visualization of M represents the columns as non-overlapping vertical streams and the rows as horizontal stripes such that the area of intersection between a column and a row is equal to the weight of the corresponding cell. To avoid large wiggle of the streams, it is desirable to keep the consecutive cells in a stream to be adjacent. Let B be the smallest axis-aligned bounding box containing the StreamTable. Then the difference between the area of B and the sum of the weights is referred to as the excess area.

We examine the complexity of optimizing various table aesthetics (minimizing excess area, or maximizing cell adjacencies in streams) in a StreamTable visualization.

- If the row permutation is fixed and the row heights are given as a part of the input, then we provide an $O(rc)$-time algorithm that optimizes these aesthetics.
- If the row permutation is fixed but the row heights can be chosen, then we discuss a technique to compute an aesthetic StreamTable by solving a quadratically constrained quadratic program, followed by iterative improvements.
- If row permutations can be chosen, then we show that it is NP-hard to find a row permutation that optimizes the area or adjacency aesthetics.

1 Introduction

Proportional area charts and cartographic visualizations commonly map data value to area. Table cartogram [4] is a brilliant way to visualize tables, where each table cell is mapped to a convex quadrilateral with area equal to the cell's weight. Furthermore, the visualization preserves cell adjacencies and the quadrilaterals are packed together in a rectangle with no empty space in between. However, since the cells in a table cartogram are represented with convex quadrilaterals, neither the rows nor the columns remain axis aligned (e.g., see Figure 4 in [4]). This motivated us to look for solutions where the rows are represented with fixed horizontal stripes and the cells are represented with axis aligned rectangles.

Streamgraphs are examples where the columns can be thought of as vertical stripes. Given a set of variables, a *streamgraph* visualizes how the value changes over time by representing each variable with an x-monotone flowing river-like stream. The width of the stream at a timestamp is determined by the value of the variable at that time. Figure 1(a) illustrates a streamgraph with five variables. Streamgraphs are often used to create infographics of temporal data [2], e.g., box office revenues for movies [1], various statistics or demographics of a population over time [7], etc.

In this paper, we introduce StreamTable that extends this idea of a streamgraph to visualize tables or spreadsheets. We formally define a StreamTable as follows.

* Work is supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), and by two CFREF grants coordinated by GIFS and GIWS.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

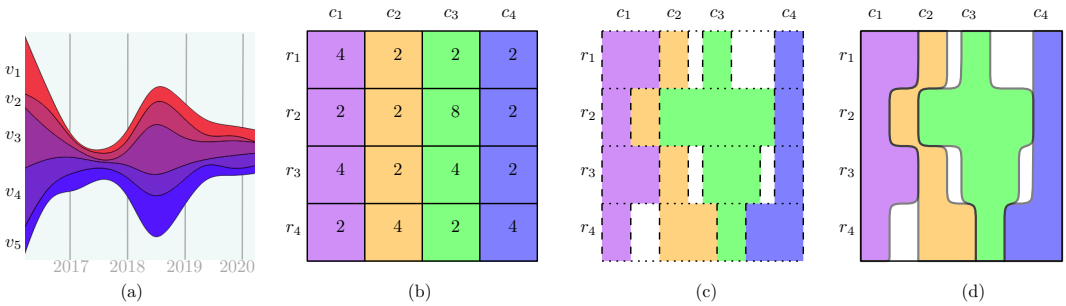


Figure 1 (a) A streamgraph. (b) A table T. (c) A StreamTable for T. (d) A StreamTable visualization with smooth streams.

StreamTable Let T be an $r \times c$ table with r rows and $c \geq 2$ columns, where each cell is weighted by a nonzero positive integer. A *StreamTable* visualization of T is a partition of an axis-aligned rectangle R into r consecutive horizontal stripes that represent the rows of T, where each stripe is further partitioned into rectangles to represent the cells of its corresponding row. A column q of T is thus represented by a sequence of rectangles corresponding to the cells of q. By a *stream* we refer to such a sequence of rectangles that represents a column of T. Furthermore, a StreamTable must satisfy the following properties.

P_1 The left side of the leftmost stream (resp., the right side of the rightmost stream) must be aligned to the left side (resp., right side) of R.

P_2 For each cell of T, the area of its corresponding rectangle in the StreamTable must be equal to the cell's weight.

Property P_1 provides an aesthetic alignment with the row labels and a sense of total visualization area. Property P_2 provides an area proportional representation of the table cells. Figure 1(b) illustrates a table and Figure 1(c) illustrates a corresponding StreamTable. The stripes (rows) are shown in dotted lines and the partition of the stripes are shown in dashed lines. Figure 1(d) illustrates an aesthetic visualization of the streams after smoothing the corners. StreamTable computation can also be viewed as a variant of floorplanning [3, 10].

Note that a StreamTable may contain rectangular regions that do not correspond to any cell. We refer to such regions as *empty regions* and the sum of the area of all empty regions as the *excess area*. While computing a StreamTable, a natural optimization criteria is to minimize this excess area. However, minimizing excess area may sometimes result into disconnected streams, e.g., Figure 2(b) illustrates a StreamTable where the consecutive rectangles for column c_2 are not adjacent. If a pair of cells are consecutive in a column but the corresponding rectangles are nonadjacent in the stream, then they *split* the stream. To maintain the stream connectedness, it is desirable to minimize the number of such splits. As illustrated in Figure 2(c)-(d), one may choose non-uniform row heights or reorder the rows to optimize the aesthetics. Such reordering operations also appear in matrix reordering problems [8] where the goal is to reveal clusters in matrix data.

Our Contribution. We explore StreamTable from a theoretical perspective and consider the following two problems.

Problem 1 (StreamTable with no Split, Minimum Excess Area, and Fixed Row Ordering). Given an $r \times c$ table T, can we compute a StreamTable for T in polynomial time with no split and minimum excess area? Note that in this problem, the StreamTable must respect the row ordering of T.

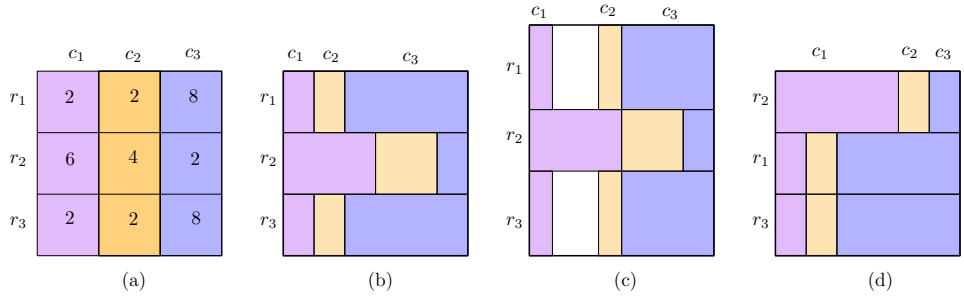


Figure 2 (a) A table. (b) A StreamTable with no excess area and 2 splits. (c) A StreamTable with non-uniform row heights, non-zero excess area, but no split. (d) A StreamTable with no excess area and 1 split, which is obtained by reordering the rows.

While Problem 1 remains open, if the input additionally specifies a set $\{h_1, \ldots, h_r\}$ of nonzero positive numbers to be chosen as row heights, then we can compute a StreamTable with minimum excess area in $O(rc)$ time. We show how to minimize the excess area further by leveraging a quadratically constrained quadratic program, and then iteratively adjusting the row heights. Since choosing a fixed row height helps to obtain a fast algorithm and to compare the cell areas more accurately, we examined whether one can leverage the row ordering to further improve the StreamTable aesthetics.

Problem 2 (Row-Permutable StreamTable with Uniform Row Heights). Given a table T and a non-zero positive number $\delta > 0$, can we compute a StreamTable in polynomial time by setting δ as the row height, and minimizing the excess area (or, the number of splits)? Note that in this problem, the row ordering can be chosen.

We show that Problem 2 is NP-hard. In particular, we show that computing a StreamTable with no excess area and minimum number of split is NP-hard and similarly, computing a StreamTable with no split and minimum excess area is NP-hard.

2 StreamTable (No Split, Min. Excess Area, Fixed Row Ordering)

In this section we compute StreamTables by respecting the row ordering of the input table. We first explore the case when the row heights are given, and then the case when the row heights can be chosen.

2.1 Fixed Row Heights

Let T be an $r \times c$ table and let $\{h_1, \ldots, h_r\}$ be a set of nonzero positive numbers to be chosen as row heights. We now introduce some notation for the rectangles and streams in the StreamTable. Let $w_{i,j}$ be the weight for the (i, j)th entry of T, where $1 \leq i \leq r$ and $1 \leq j \leq c$, and let $R_{i,j}$ be the rectangle with height h_i and width $(w_{i,j}/h_{i,j})$. Let $a_{i,j}$ and $b_{i,j}$ be the x-coordinates of the left and right side $R_{i,j}$.

We now show that a StreamTable $\mathcal{R}$ for T with no split and minimum excess area can be constructed using a greedy algorithm $\mathcal{G}$, as follows:

Step 1. Draw the rectangles $R_{i,1}$ of the first column such that they are left aligned.
Step 2. For each $j < c$, draw the jth stream by minimizing the sum of x-coordinates $a_{i,j}$, but ensuring that the stream remains connected.

Step 3. Draw the rectangles $R_{i,c}$ of the last column by minimizing the maximum x-coordinate over $b_{i,c}$, but ensuring that the rectangles are right aligned.

For every column j, let $A(\mathcal{R}, j)$ be the orthogonal polygonal chain determined by the left side of $R_{i,j}$. Similarly, we define (resp., $B(\mathcal{R}, j)$) for the right side of $R_{i,j}$. We now have the following lemma.

▶ **Lemma 2.1.** *$\mathcal{G}$ computes a StreamTable $\mathcal{R}$ with no split and minimum excess area.*

Proof. We employ an induction on the number of columns. For an $r \times c$ table T with $c = 2$, it is straightforward to verify the lemma. We now assume that the lemma holds for every table with j columns where $1 \leq j < c$. Consider now a table with c columns and let $\mathcal{R}^*$ be an optimal StreamTable with no split and minimum excess area.

We first show that the first two streams of $\mathcal{R}^*$ can be replaced with the corresponding streams of $\mathcal{R}$. To observe this first note that the stream for the first column must be drawn left-aligned, and since the rectangle heights are given, the right side of the streams $B(\mathcal{R}, 1)$ must coincide with $B(\mathcal{R}^*, 1)$. Consider now the left sides of the second streams. If $A(\mathcal{R}, 2)$ does not coincide with $A(\mathcal{R}^*, 2)$, then there must be non-zero area between them. Let A be an orthogonal polygonal chain constructed by taking the left envelope of these two chains. In other words, for each row, we choose the part of the chain that have the minimum x-coordinate. Since the streams for $\mathcal{R}$ and $\mathcal{R}^*$ are connected, the stream determined by A must be connected. Since the sum of x-coordinates is smaller for A, the polygonal chain $A(\mathcal{R}, 2)$ must coincide with A. Thus the right side of the stream, i.e., the polygonal chain $B(\mathcal{R}, 2)$, must remain to the left of $B(\mathcal{R}^*, 2)$.

We can now construct an $r \times (c-1)$ table T' by treating the polygonal chain $B(\mathcal{R}, 2)$ as $B(\mathcal{R}, 1)$. By induction, $\mathcal{G}$ provides a StreamTable $\mathcal{R}'$ with no split and minimum excess area. We can thus obtain the StreamTable $\mathcal{R}$ by replacing the first stream with the two streams that were constructed using the greedy approach. ◀

We now have the following theorem. We omit the proof due to space constraints.

▶ **Theorem 2.2.** *Given an $r \times c$ table T and a height for each row, a StreamTable $\mathcal{R}$ for T with no split and minimum excess area can be computed in $O(rc)$ time such that $\mathcal{R}$ respects the row ordering of T.*

We now show how to formulate a system of linear equations to compute a StreamTable for T with no split and minimum excess area such that the height of the ith row is set to h_i, where $1 \leq i \leq r$. This will be useful for the subsequent section. Let $d_{i,j}$ be a variable to model the adjacency between $R_{i,j}$ and $R_{i+1,j}$, where $1 \leq i \leq r-1$ and $1 \leq j \leq c$. We minimize the excess area: $\sum_{j=1}^{r} \sum_{k=1}^{c-1} h_j(a_{j,k+1} - b_{j,k})$, subject to the following constraints.

1. $a_{j,1} = a_{j+1,1}$ and $b_{j,c} = b_{j+1,c}$, where $j = 1, \ldots, r-1$. This ensures StreamTable property P_1.
2. $b_{j,k} - a_{j,k} = (w_{j,k}/h_j)$, where $j = 1, \ldots, r$ and $k = 1, \ldots, c$. This ensures property P_2.
3. $a_{j,k} \leq d_{j,k} \leq b_{j,k}$ and $a_{j+1,k} \leq d_{j,k} \leq b_{j+1,k}$, where $1 \leq j \leq r-1$ and $1 \leq k \leq c$. This ensures that there is no split in the streams.

Since $h_1, \ldots, h_r$ are fixed constants, the above system with the constraint that the variables must be non-negative can be modeled as a linear program, e.g., see Figure 3 (left).

2.2 Variable Row Heights

A straightforward solution in this case is to treat $h_1, \ldots, h_j$ as variables, which yields a quadratically constrained quadratic program. Note that scaling down the height of a StreamTable by some $\delta \in (0, 1]$ and scaling up the width by $1/\delta$ do not change the excess area. Therefore, a non-linear program solver may end up generating a final table with bad aspect ratio. Hence we suggest to add another constraint: $h_1 + \ldots + h_k = H$, where H is the desired height of the visualization. Figure 3 (right) illustrates an example where the solution (not necessarily optimal) is computed using a non-linear program solver Gurobi [6], and Figure 4 is obtained by smoothing the corners of the streams.

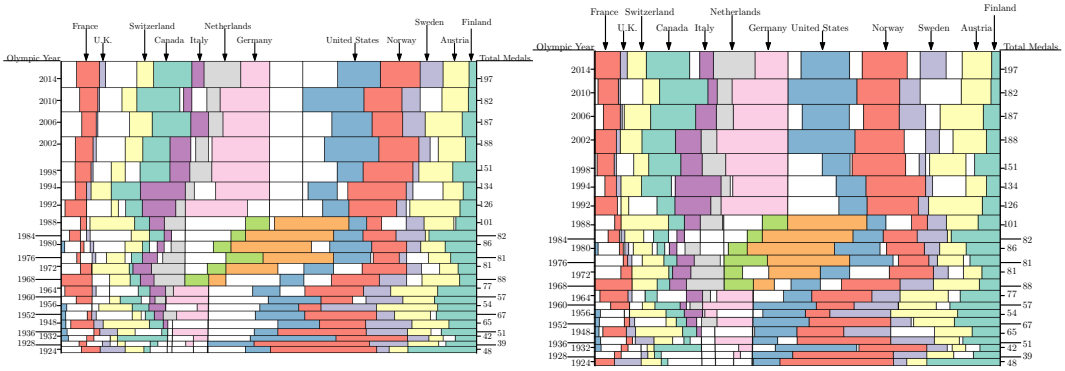


Figure 3 StreamTables of a Winter Olympics dataset (left) using a linear program with row height proportional to the row sum, and (right) using Gurobi with a fixed total height.

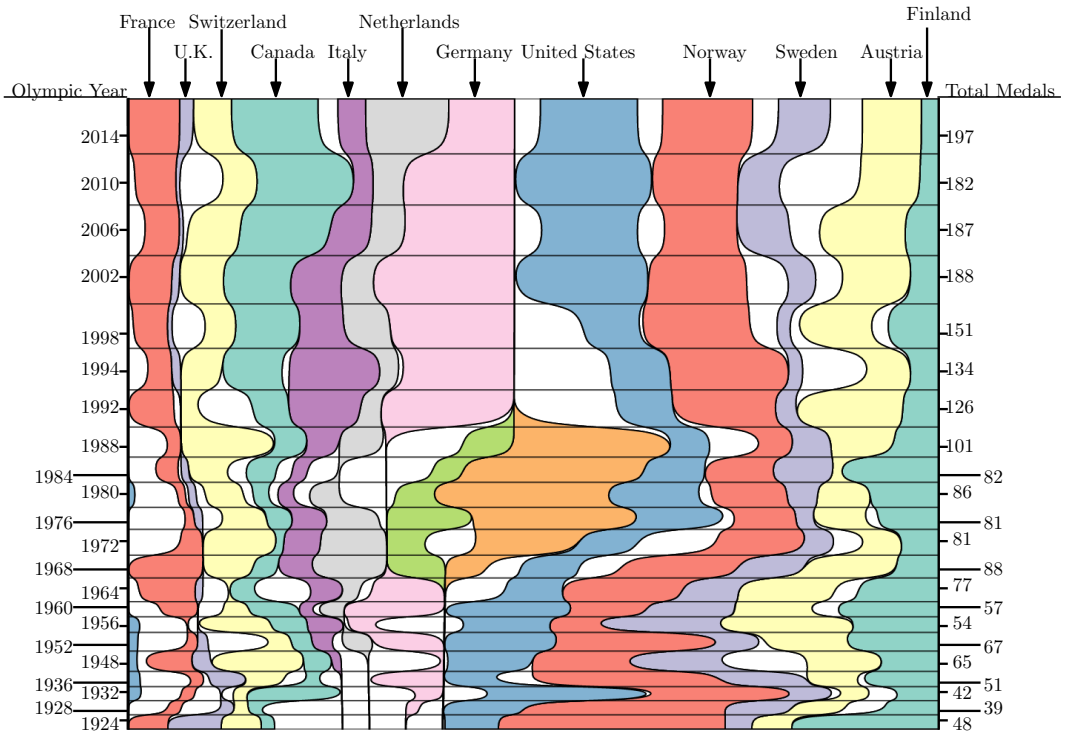


Figure 4 An aesthetic StreamTable created by corner smoothing.

We now show how a non-optimal StreamTable may be improved further by examining each empty cell individually, while deciding whether that cell can be removed by shrinking the height of the corresponding row. By $E_{i,j}$ we denote the empty rectangle between the rectangles $R_{i,j}$ and $R_{i,j+1}$. We first refer the reader to Figure 5(a)–(b). Assume that we want to decide whether the empty cell $E_{i,j}(= E_{2,4})$ can be removed by scaling down the height of the second row. The idea is to grow the rectangles to the left (resp., right) of $E_{i,j}$ towards the right (resp., left) respecting the adjacencies and area.

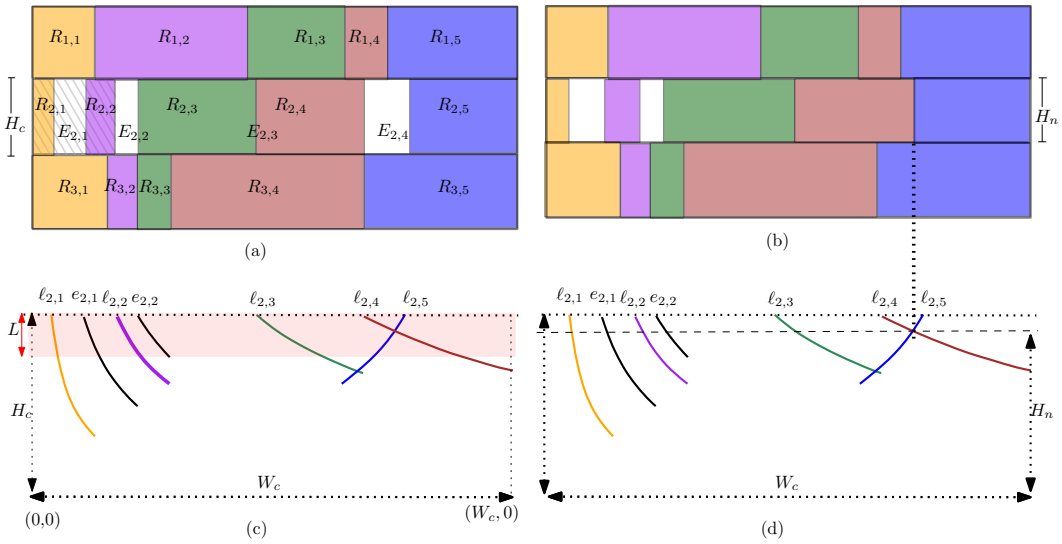


Figure 5 (a) A StreamTable with width W_c and height H_c. (b) Removal of the empty rectangle $E_{2,4}$ (c)–(d) Illustration for computing the new height H_n of the second row.

Now consider a rectangle $R_{i,k}(= R_{2,2})$ before $E_{i,j}(= E_{2,4})$. Let $G_{i,k}$ be the rectangle determined by the ith row with left and right sides coinciding with the left and right sides of $R_{i,1}$ and $R_{i,k}$, respectively. Figure 5(a) shows $G_{2,2}$ in a falling pattern. Let $\ell_{i,k}$ be the width of $G_{i,k}$. Let $A_{i,k}$ be the initial area of $G_{i,k}$, and our goal is to keep this area fixed as we scale down the height of the ith row. The height of $G_{i,k}$ is defined by $f(\ell_{i,k}) = A_{i,k}/\ell_{i,k}$. Since the rectangles of the $(i-1)$th and $(i+1)$th rows do not move, $f(\ell_{i,k})$ does not split the $(k+1)$th stream as long as $\ell_{i,k}$ is upper bounded by the right sides of $R_{i-1,k+1}$ and $R_{i+1,k+1}$. Figure 5(c) plots these functions, where H_c is the current height of the second row. The height function for $G_{2,2}$ is drawn in thick purple in the interval $[\ell_{2,2}, \min\{q_{1,3}, q_{3,3}\}]$, where $q_{1,3}$ and $q_{3,3}$ are the right sides of $R_{1,3}$ and $R_{3,3}$, respectively.

We construct such functions also for all the empty rectangles $E_{i,q}$, where $1 \leq q < j$. These are labelled with $e_{i,q}$. Finally, we construct these functions symmetrically for the rectangles that appear after $E_{i,j}$. We then find a height H_n by determining the common interval L where all these functions are valid individually (Figure 5(c)), and determining the first intersection (if any) in this interval, as illustrated in Figure 5(d).

We iterate over the empty rectangles as long as we can find an empty rectangle to improve the solution, or to a maximum number of iterations.

3 StreamTable (Uniform Row Heights, Variable Row Ordering)

We now show that computing StreamTables with no split (resp., minimum excess area) while minimizing the excess area (resp., number of splits) by reordering the rows is NP-hard.

▶ **Theorem 3.1.** *Given a table T and a non-zero positive number $\delta > 0$, it is NP-hard to compute a StreamTable with no split and minimum excess area, where each row is of height δ and the ordering of the rows can be chosen.*

Proof. We reduce the NP-complete problem *betweenness* [9]. Given an instance S of betweenness, we construct an $r \times (4c + 1)$ table T and a positive integer δ such that there exists a StreamTable for T with no split and excess area at most $\frac{15rc}{12}$ if and only if S admits a total order. We omit the details due to space constraints. ◀

▶ **Theorem 3.2.** *Given a table T and a non-zero positive number $\delta > 0$, it is NP-hard to compute a StreamTable with zero excess area and minimum number of splits, where each row is of height δ and the ordering of the rows can be chosen.*

Proof. We reduce the NP-complete problem *Hamiltonian path in a cubic graph* [5]. Given an instance G with n vertices and m edges, we construct an $n \times m$ table T and a positive integer δ such that there exists a StreamTable for T with δ height for each row, zero excess area, and at most $4(n - 1)$ splits, if and only if G admits a Hamiltonian path. We omit the details due to space constraints. ◀

References

1. Marco Di Bartolomeo and Yifan Hu. There is more to streamgraphs than movies: Better aesthetics via ordering and lassoing. *Comput. Graph. Forum*, 35(3):341–350, 2016.
2. Lee Byron and Martin Wattenberg. Stacked graphs - geometry & aesthetics. *IEEE Trans. Vis. Comput. Graph.*, 14(6):1245–1252, 2008.
3. Temo Chen and Michael K. H. Fan. On convex formulation of the floorplan area minimization problem. In Majid Sarrafzadeh, editor, *Proceedings of the 1998 International Symposium on Physical Design (ISPD)*, pages 124–128. ACM, 1998.
4. William S. Evans, Stefan Felsner, Michael Kaufmann, Stephen G. Kobourov, Debajyoti Mondal, Rahnuma Islam Nishat, and Kevin Verbeek. Table cartogram. *Comput. Geom.*, 68:174–185, 2018.
5. M. R. Garey, David S. Johnson, and Robert Endre Tarjan. The planar Hamiltonian circuit problem is NP-complete. *SIAM J. Comput.*, 5(4):704–714, 1976.
6. LLC Gurobi Optimization. Gurobi optimizer reference manual, 2020. URL: `https://www.gurobi.com/wp-content/plugins/hd_documentations/documentation/9.1/refman.pdf`.
7. Susan Havre, Elizabeth G. Hetzler, Paul Whitney, and Lucy T. Nowell. Themeriver: Visualizing thematic changes in large document collections. *IEEE Trans. Vis. Comput. Graph.*, 8(1):9–20, 2002.
8. Erkki Mäkinen and Harri Siirtola. Reordering the reorderable matrix as an algorithmic problem. In Michael Anderson, Peter C.-H. Cheng, and Volker Haarslev, editors, *Proc. of the First International Conference on Theory and Application of Diagrams*, volume 1889 of *LNCS*, pages 453–467. Springer, 2000.
9. Jaroslav Opatrny. Total ordering problem. *SIAM J. Comput.*, 8(1):111–114, 1979.
10. E. Rosenberg. Optimal module sizing in VLSI floorplanning by nonlinear programming. *ZOR Methods Model. Oper. Res.*, 33(2):131–143, 1989.

An example of a randomized order-dependent time analysis in incremental construction*

Kolja Junginger[1] and Evanthia Papadopoulou[2]

1,2 Faculty of Informatics, Università della Svizzera italiana
{kolja.junginger, evanthia.papadopoulou}@usi.ch

Abstract

Abstract Voronoi-like diagrams were introduced by the authors in SoCG 2018 [4] serving as intermediate structures in a simple randomised incremental algorithm to perform site-deletion in abstract Voronoi diagrams in expected linear time. The intermediate Voronoi-like structures depend on the permutation order of the randomized algorithm and this complicates the time complexity analysis of the incremental construction. In this abstract we present a method to perform the time-complexity analysis, which can be of independent interest when analysing the expectation of order-dependent structures.

Related Version arXiv:1803.05372v2

1 Introduction

We present the time complexity analysis of a simple randomized incremental algorithm, which was described by the authors in [4] to perform deletion in abstract Voronoi diagrams in expected linear time, and which also applies to other related Voronoi structures [5]. The technique uses *Voronoi-like diagrams* as intermediate structures, which are defined as graphs on the arrangement of the underlying bisector system. These intermediate structures, however, depend on the randomization order of the incremental algorithm and this complicates the algorithm's time complexity analysis. We present a strategy of how to perform this analysis, which can be of independent interest when analyzing expectation in order-dependent environments.

Backwards analysis [8] offers simple and elegant means to analyse a randomized incremental algorithm. It was first used by Chew [1] in a simple incremental technique to compute the Voronoi diagram of points in convex position in expected linear time. Seidel [8] demonstrated a variety of problems, whose time analysis can be performed in simple terms by backwards analysis and since then it has become a standard in computational geometry, see, e.g., [2] and references therein. He also pointed out a negative example of an order-dependent triangulation where the standard arguments were not applicable. Similarly, standard arguments are not easy to apply to our order-dependent Voronoi-like structures.

In [4], the cost of one insertion operation is expressed in terms of the resulting structure, as typically done in backwards analysis. However, depending on the permutation order, at any step i of the incremental algorithm, there can be a large number of different resulting structures, which are defined on the same set of i objects, preventing the use of standard arguments in deriving the expectation. In this paper, we give an alternative derivation. We consider all possible permutations on i objects, and partition them into disjoint groups of i permutations each. The i permutations within one group all have a different i^{th} element, while the order of the remaining elements is kept intact. We show that the step i of the

* Supported in part by the Swiss National Science Foundation, DACH project SNF-200021E_154387.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

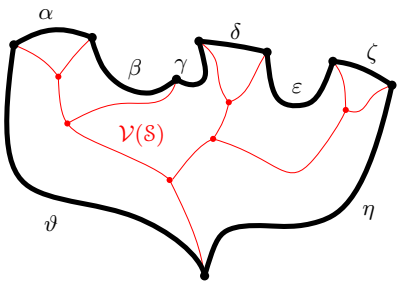


Figure 1 $\mathcal{S} = \partial\mathrm{VR}(s, S)$ shown in black and $\mathcal{V}(\mathcal{S}) = \mathcal{V}(S \setminus \{s\}) \cap \mathrm{VR}(s, S)$ shown in red; $\mathcal{S} = (\alpha, \beta, \gamma, \delta, \varepsilon, \zeta, \eta, \vartheta)$. (Reproduced from [4].)

algorithm on an entire group can be performed in total $O(i)$ time. Since all permutations are equally likely, we can derive that step i is performed in expected $O(1)$ time. The method gives a simple alternative to backwards analysis, applicable to both order-dependent and order-independent structures.

We first review concepts of abstract Voronoi and Voronoi-like diagrams and the randomized incremental construction of [4]. Then in Section 3, we give the strategy to perform the time complexity analysis, and in Section 4 we outline its derivation.

2 Review of abstract Voronoi and Voronoi-like diagrams

Let S be a set of n abstract *sites* and let $\mathcal{J} = \{J(p, q) : p \neq q \in S\}$ be their underlying bisector system, which is *admissible*, i.e., it satisfies the axioms of abstract Voronoi diagrams for every subset $S' \subseteq S$. That is, each bisector curve is an unbounded Jordan curve; each Voronoi region is non-empty and connected; Voronoi regions cover the plane; and any two bisectors intersect transversally and in a finite number of points [6].

The bisector $J(p, q)$ of two sites $p, q \in S$ is an unbounded Jordan curve that divides the plane into two open domains: the *dominance region of* p, $D(p, q)$, and the *dominance region of* q, $D(q, p)$. The *Voronoi region* of site p is

$$\mathrm{VR}(p, S) = \bigcap_{q \in S \setminus \{p\}} D(p, q).$$

The *abstract Voronoi diagram* of S is $\mathcal{V}(S) = \mathbb{R}^2 \setminus \bigcup_{p \in S} \mathrm{VR}(p, S)$.

Without loss of generality, we restrict all computations within the domain D_Γ enclosed by a large closed curve Γ (e.g., a circle or rectangle) which encloses all intersections of the bisector system. Each bisector crosses Γ twice and transversally.

Consider a Voronoi region $\mathrm{VR}(s, S)$ and let $\mathcal{S}$ denote the sequence of Voronoi edges along $\partial\mathrm{VR}(s, S)$, i.e., $\mathcal{S} = \partial\mathrm{VR}(s, S) \cap D_\Gamma$. The arcs in $\mathcal{S}$ can be interpreted as sites that induce the Voronoi diagram $\mathcal{V}(\mathcal{S}) = \mathcal{V}(S \setminus \{s\}) \cap \mathrm{VR}(s, S) \cap D_\Gamma$, see Figure 1. We compute $\mathcal{V}(\mathcal{S})$ in order to update the Voronoi diagram after deletion of the site s. $\mathcal{V}(\mathcal{S})$ is a tree, if $\mathrm{VR}(s, S)$ is bounded, and a forest otherwise. Its regions may have multiple faces that belong to the same site; in fact, $\mathcal{S}$ is a Davenport-Schinzel sequence of order 2.

Given $\mathcal{S}' \subseteq \mathcal{S}$, we need a diagram that is related to $\mathcal{S}'$. To this aim we use the notion of a *boundary curve on* $\mathcal{S}'$ and its *Voronoi-like diagram*. For any arc $\alpha \in \mathcal{S}$, let s_α denote the site in S such that $\alpha \subseteq J(s, s_\alpha)$. Consider the set $\mathcal{J}_p$ of all bisectors related to a site $p \in S$, i.e., the set of the bisectors $J(p, \cdot)$.

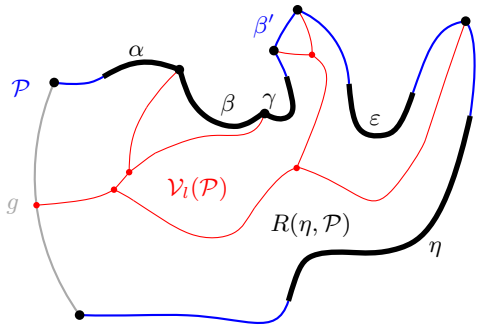


Figure 2 A boundary curve $\mathcal{P}$ on $\mathcal{S}' \subseteq \mathcal{S}$ and its Voronoi-like diagram $\mathcal{V}_l(\mathcal{P})$. $\mathcal{S}'$ is shown in bold and $\mathcal{V}_l(\mathcal{P})$ in red. The gray arc g is a Γ-arc, the blue arc β' is an auxiliary arc, and the remaining arcs are original. (Reproduced from [4].)

▶ **Definition 2.1** ([4]). A path in the arrangement of all bisectors in $\mathcal{J}_p$ is called *p-monotone*, if any two consecutive arcs α, β, with $\alpha \subseteq J(p, s_\alpha)$ and $\beta \subseteq J(p, s_\beta)$, coincide locally around their common endpoint v with the Voronoi edges of $\partial\mathrm{VR}(p, \{p, s_\alpha, s_\beta\})$, incident to v.

▶ **Definition 2.2** ([4]). A *boundary curve* $\mathcal{P}$ on a set of core arcs $\mathcal{S}' \subseteq \mathcal{S}$ is a closed s-monotone path in the arrangement of $\mathcal{J}_s \cup \Gamma$ that contains all arcs in $\mathcal{S}'$. The part of the plane enclosed by $\mathcal{P}$ is called its *domain* $D_\mathcal{P}$, see Figure 2.

A boundary curve consists of pieces of s-bisectors called *boundary arcs*, and pieces of Γ, called Γ-*arcs*. The Γ-arcs serve as links to boundary arcs and they correspond to openings of the domain $D_\mathcal{P}$ to infinity. Among the boundary arcs, those that contain an arc of $\mathcal{S}'$ are called *original* and the others are called *auxiliary*. Original arcs are expanded versions of the core arcs in $\mathcal{S}'$. In Figure 2, core arcs are shown in bold.

Let $S' \subseteq S \setminus \{s\}$ denote the set of sites that, together with s, induce the bisectors of the arcs in $\mathcal{S}'$. Let $\mathcal{J}_{S'} = \{J(p, q) \in \mathcal{J} \mid p, q \in S'\}$. We consider the arrangement of $\mathcal{J}_{S'} \cup \Gamma$.

▶ **Definition 2.3** ([4]). The *Voronoi-like diagram* of a boundary curve $\mathcal{P}$ on $\mathcal{S}' \subseteq \mathcal{S}$ is a plane graph defined on the arrangement of bisectors $\mathcal{J}_{S'}$ that induces a subdivision on the domain $D_\mathcal{P}$ as follows (see Figure 2): (1) for each boundary arc $\alpha \in \mathcal{P}$ there is exactly one distinct face $R(\alpha, \mathcal{P})$, called the *Voronoi-like region* of α, whose boundary is an s_α-monotone path in $\mathcal{J}_{S'} \cup \Gamma$ connecting the endpoints of α; and (2) the faces cover $D_\mathcal{P}$: $\bigcup_{\alpha \in \mathcal{P} \setminus \Gamma} \overline{R(\alpha, \mathcal{P})} = \overline{D_\mathcal{P}}$. The Voronoi-like diagram of $\mathcal{P}$ is $\mathcal{V}_l(\mathcal{P}) = D_\mathcal{P} \setminus \bigcup_{\alpha \in \mathcal{P}} R(\alpha, \mathcal{P})$.

$\mathcal{V}_l(\mathcal{P})$ is unique and its complexity is $O(|\mathcal{P}|)$, where $|\mathcal{P}|$ is the number of boundary arcs in $\mathcal{P}$ [4].

The incremental construction. Any random permutation of the arcs in $\mathcal{S}$, defines a series of boundary curves $\mathcal{P}_i, i = 1, \ldots, h$, $h = |\mathcal{S}|$, and a series of shrinking domains $D_{\mathcal{P}_i}$, where $\mathcal{P}_1$ is the boundary curve defined by a single core arc, and $\mathcal{P}_h = \partial(\mathrm{VR}(s, S) \cap D_\Gamma)$; $\mathcal{V}_l(\mathcal{P}_h) = \mathcal{V}(\mathcal{S})$ [4]. The incremental algorithm is inspired by Chew [1] and works in two phases for a random permutation of $\mathcal{S}$ $o = (\alpha_1, \ldots, \alpha_h)$. In phase 1, delete the arcs in $\mathcal{S}$ in the reverse order o^{-1}, while registering their neighboring arcs at the time of deletion. In phase 2, insert back the arcs one by one in the order o, starting at $\mathcal{P}_1 = \partial(D(s, s_{\alpha_1}) \cap D_\Gamma)$ and $\mathcal{V}_l(\mathcal{P}_1) = \emptyset$. At any step i, we compute $\mathcal{V}_l(\mathcal{P}_{i+1})$ from $\mathcal{V}_l(\mathcal{P}_i)$ by inserting the core arc α_i following an *arc insertion operation* $\oplus$, which is detailed in [4]. The insertion point for α_i is determined by the recorded neighbors from phase 1, followed by a scan of any auxiliary arcs between them.

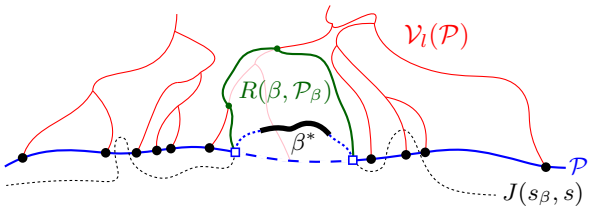


Figure 3 $\mathcal{V}_l(\mathcal{P}_\beta)$ is derived from $\mathcal{V}_l(\mathcal{P})$ by inserting the region $R(\beta, \mathcal{P}_\beta)$ [4].

The arc insertion operation $\oplus$. Given $\mathcal{V}_l(\mathcal{P})$ and $\beta^* \in \mathcal{S}'$, such that $\beta^* \notin \mathcal{P}$, we identify the original arc $\beta \in J(s, s_\beta) \cap D_\mathcal{P}$ such that $\beta \supseteq \beta^*$, and insert it in $\mathcal{P}$ to obtain $\mathcal{P}_\beta = \mathcal{P} \oplus \beta$ and $\mathcal{V}_l(\mathcal{P}_\beta) = \mathcal{V}_l(\mathcal{P}) \oplus \beta$. To derive $\mathcal{P}_\beta$ we substitute the portion of $\mathcal{P}$ between the endpoints of β, with β, see Figure 3. Then, a *merge curve* $J(\beta)$ is computed, similarly to an ordinary Voronoi diagram, which connects the endpoints of β and reveals the Voronoi-like region $R(\beta, \mathcal{P}_\beta)$, such that $J(\beta) \cup \beta = \partial R(\beta, \mathcal{P}_\beta)$. Updating the diagram, we obtain $\mathcal{V}_l(\mathcal{P}) \oplus \beta$, which turns out to be $\mathcal{V}_l(\mathcal{P}_\beta)$. The case analysis and correctness were established in [4]. The time complexity is proportional to the complexity of $J(\beta)$, plus some additional parameters for scanning auxiliary arcs and for splitting Voronoi-like regions. For the purposes of this paper, we ignore these additional parameters and we reduce the dependency of the time complexity to $|R(\beta, \mathcal{P}_\beta)|$.

In summary, the simplified cost of the algorithm's step i in this paper is assumed $|R(\alpha_i, \mathcal{P}_{i+1})|$.

3 The time analysis strategy

Consider the *decision tree* $\mathcal{T}$, which encodes all possible random choices that can be made by our incremental algorithm on a set of h core arcs $\mathcal{S}$, $h = |\mathcal{S}|$, see Fig. 4. Any path in $\mathcal{T}$ from the root to a leaf corresponds to one possible execution of the incremental construction. At level-i, there are $h!/(h-i)!$ nodes, and each node corresponds to a unique permutation of i core arcs. The arity of each node is $h - i$ corresponding to all possible choices of the algorithm at this node. $\mathcal{T}$ has $h!$ leaves and its root corresponds to the empty permutation.

Let $\mathcal{S}_i \subseteq \mathcal{S}$ be a subset of i core arcs. $\mathcal{S}_i$ is associated with $i!$ different nodes at level-i of $\mathcal{T}$, which are called the *block* of the set $\mathcal{S}_i$. Each node within a block is associated with a boundary curve and its diagram. The boundary curves of different nodes in the same block can vary considerably depending on the path, i.e., the permutation order, that leads to each

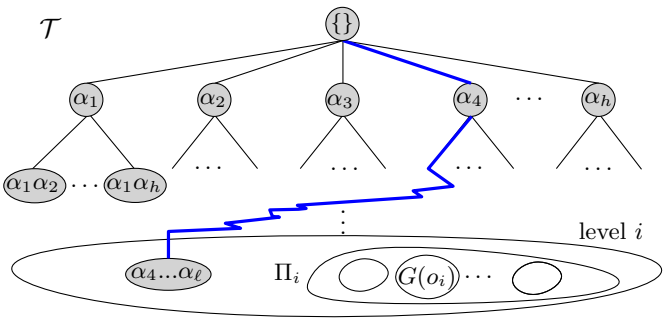


Figure 4 There are $h!/(h-i)!$ nodes at level-i of the decision tree $\mathcal{T}$, each corresponding to a unique permutation of i core arcs. Level i is partitioned into groups of size i.

node. $\mathcal{T}$ has $\binom{h}{i}$ distinct such blocks at level i.

We use the following strategy: we partition each block of nodes at level-i into $(i-1)!$ disjoint groups of i nodes each; for each group we show that the step i of our algorithm requires total time $O(i)$, for all the i permutations of the group.

Let $o_i = (\alpha_1, \alpha_2, \ldots, \alpha_i)$ be an arbitrary permutation of $\mathcal{S}_i$. From o_i we define a group $G = G(o_i)$ of i permutations: for each $1 \leq j < i$, remove α_j from its position in o_i and append it to the end of o_i.

$$o_i = (\alpha_1, \alpha_2, \ldots, \alpha_{j-1}, \boxed{\alpha_j}, \alpha_{j+1} \ldots, \alpha_{i-1}, \alpha_i) \tag{1}$$

$$o_j = (\alpha_1, \alpha_2, \ldots, \alpha_{j-1}, \quad \alpha_{j+1}, \ldots, \alpha_{i-1}, \alpha_i, \boxed{\alpha_j}) \tag{2}$$

Each permutation o_j in G corresponds to a boundary curve $\mathcal{B}_j$, $1 \leq j \leq i$, which is derived by arc insertion following the order in o_j. $\mathcal{B}_i$ is the base boundary curve of the group that is derived from o_i. Figure 5 illustrates an example for $i = 3$.

The rule of equation (2) follows two principles: 1) each of the i elements in $\mathcal{S}_i$ appears in the i^{th} position of exactly one permutation in each group; and 2) each permutation has a minimal number of *inversions* with respect to o_i. Property 1 is used to derive the time complexity of step i on G by reducing it to the structural complexity of a resulting diagram. Property 2 minimizes the differences between the resulting boundary curves. By combining the two, we can reduce the time complexity of step i to the structural complexity of $\mathcal{V}_l(\mathcal{B}_i)$.

Let $T(i, o_j)$ denote the time complexity of step i when inserting the last arc of permutation o_j in deriving $\mathcal{V}_l(\mathcal{B}_j)$.

▶ **Lemma 3.1.** *The time for step i on the entire group $G = G(o_i)$ is*

$$T(i, G) = \sum_{o_j \in G} T(i, o_j) = O(i)$$

After proving Lemma 3.1 it remains to argue that the partitioning of each block of $i!$ nodes (permutations) following the scheme of equation (2) is possible. The answer to this question is provided by Levenshtein [7] and this was pointed out to us by Stefan Felsner [3].

▶ **Lemma 3.2.** *Let Π_i denote the block of all $i!$ permutations of the set $\mathcal{S}_i$. There exists a set $F \subset \Pi_i$ of $(i-1)!$ permutations such that $\Pi_i = \dot{\bigcup}_{o \in F} G(o)$.*

Levenshtein calls this set F a *code capable of correcting single deletions* and proves that these codes exist for all $i \in \mathbb{N}$ [7, Theorem 3.1]. Given Lemmata 3.1,3.2, we conclude:

▶ **Theorem 3.3.** *The expected time complexity of step i of the randomized algorithm is $O(1)$.*

4 Proving Lemma 3.1

In this section we establish relations between the Voronoi-like diagrams of $\mathcal{B}_i$ and the rest of the boundary curves $\mathcal{B}_j$ in $G = G(o_i)$, $j < i$, so that we can prove Lemma 3.1. Figure 5 illustrates an example of $\mathcal{B}_i$, $i = 3$, and $\mathcal{B}_1$.

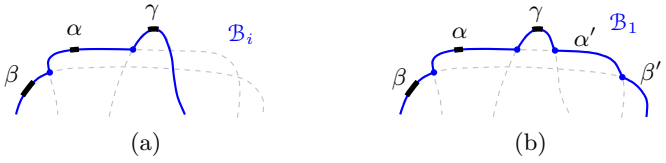


Figure 5 (a) Boundary curve $\mathcal{B}_i$, $o_i = (\gamma, \beta, \alpha)$. (b) Boundary curve $\mathcal{B}_1$, $o_1 = (\beta, \alpha, \gamma)$.

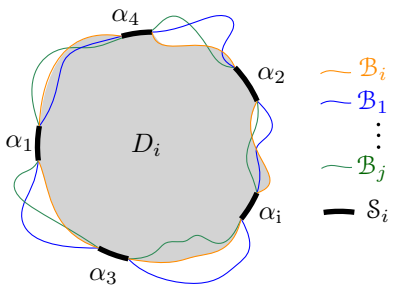


Figure 6 Schematic differences between the boundary curves $\mathcal{B}_1, \ldots, \mathcal{B}_i$.

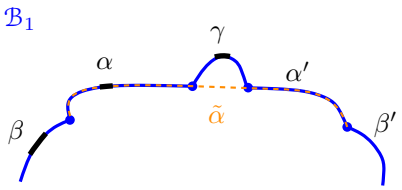


Figure 7 Let $o_1 = (\beta, \alpha, \gamma)$ and $o_i = (\gamma, \beta, \alpha)$; then $\alpha = \text{source}_j(\alpha')$. $\mathcal{B}_i$ is shown in Fig. 5(a).

Let D_i denote the domain of the base boundary curve $\mathcal{B}_i$ in $G = G(o_i)$, see Figure 6. The boundary curves $\mathcal{B}_j$, $j < i$, significantly overlap with $\mathcal{B}_i$ as they share the same set of core arcs $\mathcal{S}_i$. However, they may also get in and out of the domain D_i because their auxiliary arcs need not be the same. Let $\text{in}_j = \mathcal{B}_j \cap D_i$, and $\text{out}_j = \mathcal{B}_j \setminus \overline{D_i}$, denote the portion of $\mathcal{B}_j$ inside, and outside of D_i, respectively.

▶ **Observation 4.1.** *The boundary curve $\mathcal{B}_j$, $j \neq i$, contains no auxiliary arcs of the core arc α_j and these are the only auxiliary arcs of $\mathcal{B}_i$ that do not also appear in $\mathcal{B}_j$.*

▶ **Definition 4.2.** Let α' be an auxiliary arc in $\mathcal{B}_j$ and let $\alpha \in \mathcal{S}_i$ be a core arc of the same site. We say that α' *is an auxiliary arc of* α if there is an original arc $\tilde{\alpha} \supseteq \alpha \cup \alpha'$, which was created for the first time when inserting the core arc α during the construction of $\mathcal{B}_j$, see Figure 7. The core arc α is called the *source of* α' and is denoted as $\text{source}_j(\alpha')$. If α' appears counterclockwise (resp. clockwise) from its source α on the underlying bisector then α' is called a *ccw* (resp. *cw*) auxiliary arc.

Let in_j^+ (resp. in_j^-) include the ccw (resp. cw) auxiliary arcs of in_j. In Figure 5, arcs α' and β' belong in in_j^+ for $j = 1$.

▶ **Observation 4.3.** *Let $\alpha' \in in_j$ and let $\alpha_k = source_j(\alpha')$. Then, $k > j$, i.e., α_k follows α_j in o_i. Further, if $\alpha' \in in_j^+$ then $(\alpha_k, \alpha_j, \alpha')$ appear ccw in $\mathcal{B}_j$.*

We define the set N_j of source arcs for each j.

$$N_j = \{\text{source}_j(\alpha') \in \mathcal{S}_i \mid \alpha' \in \text{in}_j^+\}.$$

Using Observation 4.3, we derive the following disjointness property. In contrast the sets in_j^+ and in_k^+, $k \neq j$, may have many common arcs.

▶ **Lemma 4.4.** *$N_j \cap N_k = \emptyset$ for all $k \neq j$. Thus, $\sum_{j=1}^{i} |N_j| = O(i)$.*

Next, we point out the special structure of in_j^+, which is shown in Figure 8.

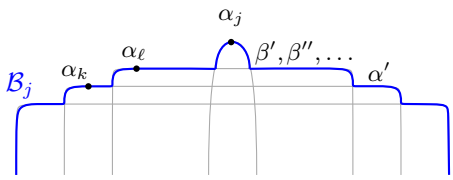


Figure 8 If $\alpha', \beta' \in \text{in}_j^+$, then $j < k < \ell$, and $(\alpha_k, \alpha_\ell, \alpha_j, \beta', \alpha')$ appear in ccw order on $\mathcal{B}_j$.

▶ **Observation 4.5.** *Let $\alpha', \beta' \in in_j^+$ such that $\alpha_k = source_j(\alpha')$, $\alpha_\ell = source_j(\beta')$, and $k < \ell$. Then, $j < k < \ell$, and $(\alpha_k, \alpha_\ell, \alpha_j, \beta', \alpha')$ appear in ccw order in $\mathcal{B}_j$. All auxiliary arcs of α_ℓ appear before the auxiliary arcs of α_k as we move on $\mathcal{B}_j$ counterclockwise from α_j.*

Next, we compare $R(\alpha_j, \mathcal{B}_j)$ and $R(\alpha_j, \mathcal{B}_i)$ and bound the differences in their adjacencies. We observe that any common arcs to both $\mathcal{B}_j$ and $\mathcal{B}_i$ that have adjacent regions in $\mathcal{V}_l(B_i)$, the same arcs must also have adjacent regions in $\mathcal{V}_l(B_j)$. We also use Observations 4.1 and 4.5.

▶ **Lemma 4.6.** $|R(\alpha_j, \mathcal{B}_j)| \leq 2|R(\alpha_j, \mathcal{B}_i)| + |N_j|$.

By Lemmata 4.6 and 4.4, we derive that $\sum_{j=1}^{i} |R(\alpha_j, \mathcal{B}_j)| = O(i)$. This completes the proof of Lemma 3.1 for the simplified time complexity formula of this abstract.

Acknowledgments. We thank Stefan Felsner for making the connection to the seemingly unrelated result of Levenshtein [7] on perfect codes, which established the proof of Lemma 3.2.

References

1 L. Paul Chew. Building Voronoi diagrams for convex polygons in linear expected time. Technical report, Dartmouth College, Hanover, USA, 1990.
2 M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 3rd edition, 2008.
3 Stefan Felsner. Personal communication, 2019.
4 Kolja Junginger and Evanthia Papadopoulou. Deletion in Abstract Voronoi Diagrams in Expected Linear Time. In *34th International Symposium on Computational Geometry (SoCG 2018)*, volume 99 of *LIPIcs*, pages 50:1–50:14, Dagstuhl, Germany, 2018. URL: `http://drops.dagstuhl.de/opus/volltexte/2018/8763`.
5 Kolja Junginger and Evanthia Papadopoulou. Deletion in abstract Voronoi diagrams in expected linear time and related problems. *CoRR*, abs/1803.05372v2, 2020. URL: `http://arxiv.org/abs/1803.05372`, `arXiv:1803.05372v2`.
6 Rolf Klein. *Concrete and Abstract Voronoi Diagrams*, volume 400 of *Lecture Notes in Computer Science*. Springer-Verlag, 1989.
7 Vladimir Levenshtein. On perfect codes in deletion and insertion metric. *Discrete Mathematics and Applications*, 2(3):241–258, 1992.
8 Raimund Seidel. Backwards analysis of randomized geometric algorithms. In *Trends in Discrete and Computational Geometry, Algorithms and Combinatorics*, volume 10, pages 37–68. Springer-Verlag, 1993.

A Tail Estimate with Exponential Decay for the Randomized Incremental Construction of Search Structures*

Joachim Gudmundsson[1] and Martin P. Seybold[1]

1 **School of Computer Science, University of Sydney, Australia**
joachim.gudmundsson@sydney.edu.au, mpseybold@gmail.com

Abstract

We revisit the randomized incremental construction (RIC) of the Trapezoidal Search DAG (TSD) for a set of n segments. It is well known that this point location structure has $\mathcal{O}(n+k)$ expected size and $\mathcal{O}(n \ln n + k)$ expected construction time, where k is the number of intersection points.

Our main result is an improved tail bound, with exponential decay, for the size of the TSD on non-crossing segments ($k = 0$): There is a constant such that the probability for a TSD to exceed its expected size by more than this factor is at most $1/e^n$. This yields improved bounds on the TSD construction and their maintenance. I.e. TSD construction takes with high probability $\mathcal{O}(n \ln n)$ time and TSD size can be made worst case $\mathcal{O}(n)$ with an expected rebuild cost of $\mathcal{O}(1)$.

1 Introduction

RIC is one of the most successful and influential paradigms in Computational Geometry. The idea is to first permute all n input objects, uniformly at random, before inserting them, one at a time, in an initially empty structure under this order. The theory developed for history based RIC lead to a tail bound technique [13, 8] that holds as soon as the actual geometric problem under consideration provides a certain boundedness property. To our knowledge, the strongest tail bound to date is from Clarkson et al. [8, Corollary 26], which states the following. Given a function M such that $M(j)$ upper bounds the size of the structure on j objects. If $M(j)/j$ is non-decreasing, then, for all $\lambda > 1$, the probability that the history size exceeds $\lambda M(n)$ is at most $(e/\lambda)^\lambda/e$. This includes the TSD size for non-crossing segments ($k = 0$). Assuming intersecting segments, Matoušek and Seidel [12] show how to use an isoperimetric inequality for permutations to derive a tail bound of $\mathcal{O}(n^{-c})$, given there are at least $k \geq Cn \log^{15} n$ many intersections in the input (both constants c and C depend on the deviation threshold λ). Mehlhorn et al. [13] show that the general approach can yield a tail bound of at most $1/e^{\Omega(k/n \ln n)}$, given there are at least $k \geq n \ln n \ln^{(3)} n$ intersections in the input. Recently, Sen [22] gave tail estimates for 'conflict graph' based RICs (cf. Chapter 3.4 in [17]) using Freedman's inequality for Martingales. The work also shows a lower bound on tail estimates for the runtime, i.e. the total number of 'conflict graph' modifications, for computing the trapezoidation of non-crossing segments that rules out high probability tail bounds [22, Section 6]. In this variation of the RIC, not only one endpoint per segment is maintained in conflict lists, but edges in a bipartite conflict graph, over existing trapezoids and uninserted segments, that contain an edge if the geometric objects intersect (see Appendix and Figure 4 in [22]). Hence this lower bound construction does not translate to the TSD (i.e. history based RIC).

* Full paper: https://arxiv.org/abs/2101.04914

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Technique	Size	With Prob. $\geq$	Condition
Isoperimetric[12]	$\mathcal{O}(k)$	$1-\mathcal{O}(1/n^c)$	$k \geq Cn\log^{15} n$
Hoeffding[13]	$\mathcal{O}(k)$	$1-1/e^{\Omega(k/n\ln n)}$	$k \geq n\ln n\ln^{(3)} n$
Freedman[22]	$\mathcal{O}(k)$	$1-1/e^{k/n\alpha(n)}$	$k \geq n\ln n$
Hoeffding[8]	$\mathcal{O}(n)$	$1-(e/\lambda)^{\lambda}/e$	
Pairwise Events	$\mathcal{O}(n)$	$1-1/e^{n}$	

Table 1 Tail bounds for the history size of TSDs on n segments. k denotes the number of intersection points and $\alpha(n)$ the inverse of Ackermann's function.

We introduce a new and direct technique to analyze the size of the TSD that is based on pairwise events and an inductive application of Chernoff's method. Our main result is a *much sharper* tail estimate for the TSD size of non-crossing segments (see Table 1). This complements the known high probability bound for the point location cost and shows that the TSD has, with very high probability, size $\mathcal{O}(j)$ after every insertion step j.

2 Recap: Trapezoidal Search DAGs

Let S be a set of n segments in the plane. We identify the permutations over S with the set of bijective mappings to $\{1,\ldots,n\}$, i.e. $\mathbf{P}(S) = \{\pi : S \to \{1,\ldots,n\} \mid \pi \text{ bijective}\}$. The integer $\pi(s)$ is called the *priority* of the segment s.

An implicit, infinitesimal shear transformation allows to assume, without loss of generality, that all distinct endpoints have different x-coordinates (e.g. Chapter 6.3 in [9]). Trapezoidation $\mathcal{T}(S)$ is defined by emitting two vertical rays (in negative and positive y-direction) from each end or intersection point until the ray meets the first segment or the bounding rectangle (see Figure 1). To simplify presentation, we also implicitly move common endpoints infinitesimal along their segment, towards their interior. This gives that non-crossing segments have *no* points in common, though there may exist some spatially empty trapezoids in $\mathcal{T}(S)$. We identify $\mathcal{T}(S)$ with the set of faces in this decomposition of the plane. Elements in $\mathcal{T}(S)$ are trapezoidal regions that have a boundary that is defined by at most four segments of S (see Figure 1). Note that boundaries of the trapezoids in $\mathcal{T}(S)$ are solely determined by the set of segments S, irrespective of the permutation. We will need the following notations. Let $\gamma > 1$ be the smallest constant[1] such that $|\mathcal{T}(S)| \leq \gamma n$ holds for any S that is sufficiently large. For a segment $s \in S$, let $f(s,S) = \{\Delta \in \mathcal{T}(S) : \Delta \text{ is bounded by } s\}$ denote the set of faces that are bounded by s (i.e. *top*, *bottom*, $left$, or $right$). Let $s_i = \pi^{-1}(i)$ be the priority i segment and let $S_{\leq k} = \{s_1,\ldots,s_k\}$.

The expected size of the TSD is typically analyzed by considering $\sum_{j=1}^{n} D_j$ where the random variable $D_j := |f(s_j, S_{\leq j})|$ denotes the number of faces that are created by inserting s_j into trapezoidation $\mathcal{T}(S_{\leq j-1})$, equivalently that are removed by deleting s_j from $\mathcal{T}(S_{\leq j})$ (see Figure 2). Classic Backward Analysis [9, p. 136] in this context is the following argument. Let $S' \subseteq S$ be a fixed subset of j segments, then

$$\mathop{\mathbb{E}}_{\mathbf{P}(S)}\left[D_j \middle| S_{\leq j} = S'\right] = \frac{1}{j}\sum_{s\in S'}\sum_{\Delta\in\mathcal{T}(S')} \chi\big(\Delta \in f(s,S')\big) \leq \frac{4\gamma j}{j}\ ,$$

where the binary indicator variable $\chi(\Delta \in f(s,S'))$ is 1 iff the trapezoid Δ is bounded by segment s. The equality is due to that every segment in S' is equally likely to be picked

[1] See, e.g., Lemma 6.2 in [9] that shows $|\mathcal{T}(S)| \leq 3|S| + 1$ for non-crossing segments.

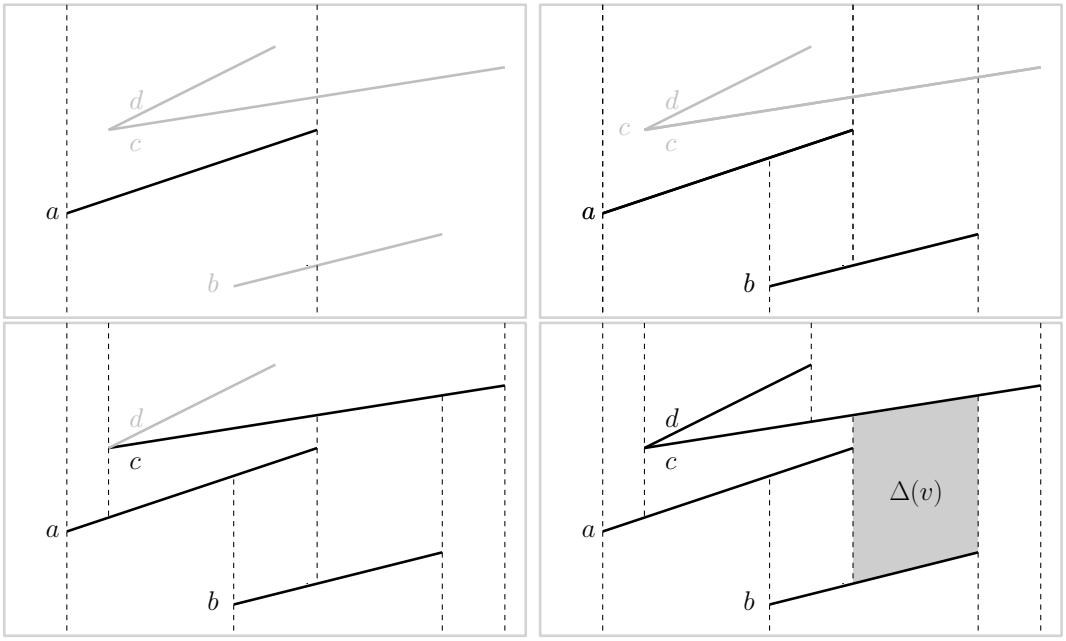


Figure 1 Trapezoidations over the segments $S = \{a = (a.l, a.r), b = (b.l, b.r), c = (c.l, c.r), d = (d.l, d.r)\}$ where $c.l = d.l$ is a common endpoint. $\mathcal{T}(\{a\})$, $\mathcal{T}(\{a, b\})$, $\mathcal{T}(\{a, b, c\})$, and $\mathcal{T}(\{a, b, c, d\})$ have 4, 7, 10, and 13 faces respectively (cf. leaves in Figure 2).

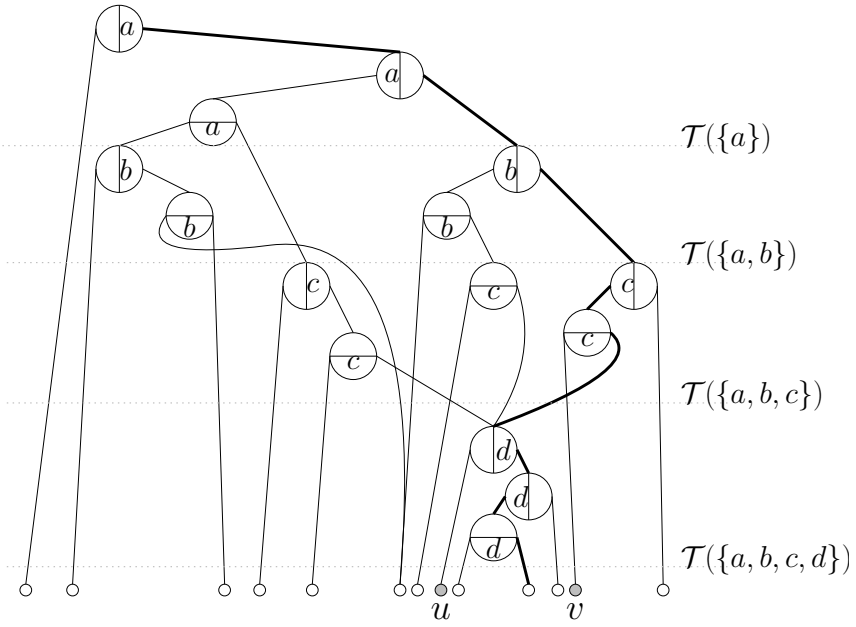


Figure 2 TSD for the history of trapezoidations under permutation $\pi = \left(\begin{smallmatrix} a & b & c & d \\ 1 & 2 & 3 & 4 \end{smallmatrix}\right)$ from Figure 1. TSD node v corresponds to the trapezoid $\Delta(v)$, which has the boundaries $top(\Delta(v)) = c$, $bottom(\Delta(v)) = b$, $left(\Delta(v)) = a.r$, and $right(\Delta(v)) = b.r$ and the spatially empty $\Delta(u)$ is due to common endpoint $left(\Delta(u)) = c.l = d.l = right(\Delta(u))$. Path with heavy line width is not a search path, since $d.r$ is left of $a.r$.

$$X(\pi) = \begin{array}{|c|c|c|c|}\hline & & & \\ \hline 1 & & & \\ \hline 1 & 1 & & \\ \hline 0 & 0 & 1 & \\ \hline \end{array} \qquad \begin{array}{ll|l} A_2 = \{a\} & N_2 = \{\} & D_1 = 4 \\ A_3 = \{a,b\} & N_3 = \{\} & D_2 = 5 \\ A_4 = \{c\} & N_4 = \{a,b\} & D_3 = 6 \\ & & D_4 = 4 \end{array}$$

Table 2 Outcome of the pairwise events and the partitions for segments $S = \{a, b, c, d\}$ and order π from Figures 1 and 2.

for s_j. Since the bound on the value of the conditional expectation does not depend on the actual set S', we have $\mathbb{E}[D_j] \leq 4\gamma$ unconditionally for each step j. Since the destruction of a face (of a leaf node) creates at most three search nodes, linearity of expectations gives that the expected number of TSD nodes is at most $12\gamma n$.

3 A tail bound using Pairwise Events

We define for each $1 \leq i < j \leq n$ an event, i.e. a binary random variable, $X_{i,j} : \mathbf{P}(S) \to \{0, 1\}$ by setting

$$X_{i,j} = \begin{cases} 1 & \text{if } f(s_j, S_{\leq j}) \text{ contains a trapezoid bounded by } s_i \\ 0 & \text{otherwise} \end{cases} .$$

To simplify presentation, we place the events in a lower triangle matrix and call the set $r(j) := \{X_{i,j} : 1 \leq i < j\}$ the events of row j and the set $c(i) := \{X_{i,j} : i < j \leq n\}$ the events of column i.

Imagine that the random permutation is built backwards, i.e. by successively choosing one of the remaining elements uniformly at random to assign the largest available priority value. For every step j at least one of the row events occurs, i.e. $0 < \sum_{i<j} X_{i,j} < j$, since at least one trapezoid is destroyed in step j and the exact probability of the events $r(j)$ depends on the geometry of the segments $S_{\leq j}$.

Consider the events in row j. Conditioned that the random permutation starts with $S' = S \setminus \{s_{j+1}, \ldots, s_n\}$, the experiment chooses $s \in S'$ uniformly at random and assigns the priority value j to it. Clearly the number of occurring (row) events depends on which segment of S' is picked as s_j, as this determines the value $f(s_j, S')$. Note that the choice of s_j also fixes a partition of $S' = \{s_j\} \cup A \cup N$ into those segments that are and aren't adjacent to s_j, the sets A and N respectively. This defines a partition $S_{\leq j} = \{s_j\} \cup A_j \cup N_j$ in every backward step j. Eventually s_i is picked from $S_{\leq i}$, which determines the outcomes of *all* column events $c(i)$. I.e. when s_i is picked from $S_{\leq i}$, the objects in the set are multi colored (A_j or N_j for each $j > i$) and $X_{i,j}$ occurs if and only if the pick has the respective color A_j.

This shows for the event probability that

$$\mathbb{E}[X_{i,j} | s_{i+1}, \ldots, s_n] = \mathbb{E}[X_{i,j} | Y, s_{i+1}, \ldots, s_n] \tag{1}$$

for every $Y \in c(i')$ with $i' > i$. See Table 2 for an example.

Moreover, we have, for every $t > 0$, the two equations

$$\mathbb{E}[\prod_{j>1} \exp(tX_{1,j}) | s_2, \ldots, s_n] = \prod_{j>1} \mathbb{E}[\exp(tX_{1,j}) | s_2, \ldots, s_n] \tag{2}$$

$$\mathbb{E}[\prod_{j>i} \exp(tX_{i,j}) | s_i, \ldots, s_n] = \prod_{j>i} \mathbb{E}[\exp(tX_{i,j}) | s_i, \ldots, s_n] \ , \tag{3}$$

where $(s_i, \ldots, s_n)$ denotes the conditioning of the random permutations on this suffix. Note that for a set of events $\{B_i\}$ that are either certain or impossible, i.e. $\mathbb{E}[B_i] = \Pr[B_i] \in \{0, 1\}$, we have that the outcome of each event is identical to its probability and thus $\mathbb{E}[\prod_i \exp(tB_i)] = \prod_i \exp(t\mathbb{E}[B_i]) = \prod_i \mathbb{E}[\exp(tB_i)]$.

There is a close relation between the row events $r(j)$ and the random variable D_j.

▶ **Lemma 3.1.** *Let S be a set of non-crossing segments. For every $\pi \in \mathbf{P}(S)$ and $j \geq 2$, we have $D_j(\pi)/6 \leq \sum_{i<j} X_{i,j}(\pi) \leq 3D_j(\pi)$.*

Proof. Let $S' := S_{\leq j}(\pi)$ be the segments with priority at most j in π. Clearly every trapezoid that is incident to s_j is bounded by at most three other segments, which gives the upper bound. For the lower bound, we first count those trapezoids of $f(s_j, S')$ that have s_j as top or bottom boundary. Let P be the set of endpoints that define the vertical boundaries of these trapezoids, excluding the endpoints of $s_j = (q_l, q_r)$. Partition P into points P^+ above and P^- below s_j, which blocks their vertical rays in trapezoidation $\mathcal{T}(S')$. Consider the two sets P^+ and P^- sorted by their x-coordinates. Between the endpoints of s_j, the vertical boundaries of points in P^+ can define at most $|P^+| + 1$ trapezoids. Hence $f(s_j, S')$ contains at most $|P| + 2$ trapezoids that have s_j on their top or bottom boundary. The remaining trapezoids of $f(s_j, S')$ are either bounded by q_l or by q_r. There is at most one trapezoid in $\mathcal{T}(S')$ that has endpoint q_l as right vertical boundary but not s_j as bottom or top segment. The argument for q_r is symmetric.

Putting the bounds for all cases of trapezoids in $f(s_j, S')$ together and using that $|P| \leq 2\sum_{i<j} X_{i,j}(\pi)$, we have

$$D_j(\pi) \leq (2 + |P|) + 2 \leq \left(2 + 2\sum_{i<j} X_{i,j}(\pi)\right) + 2 \leq 6\sum_{i<j} X_{i,j}(\pi) \ .$$

In the last step we used the fact that $1 \leq \sum_{i<j} X_{i,j}(\pi)$. ◀

Hence $\sum_j D_j(\pi)/6 \leq \sum_{i,j} X_{i,j}(\pi) \leq 3\sum_j D_j(\pi)$ holds for every permutation $\pi \in \mathbf{P}(S)$. This also gives, for every j, the bounds on the expected values

$$\mathbb{E}\left[D_j\right]/6 \quad \leq \quad \mathbb{E}\left[\sum_{i<j} X_{i,j}\right] \quad \leq \quad 3\,\mathbb{E}\left[D_j\right] \quad \leq \quad 12\gamma \ ,$$

and thus $\mathbb{E}[\sum_{i,j} X_{i,j}] \leq 12\gamma n$.

Furthermore, consider the isolated event $X_{i,j}$ in row j. Since the element s_i is picked uniformly at random from the set $S_{<j}$, we have that its event probability is within the range

$$\frac{1/6}{j-1} \ \leq \ \mathbb{E}[X_{i,j}] \ \leq \ \frac{12\gamma}{j-1} \ . \tag{4}$$

Hence the events have roughly Harmonic distribution, i.e. up to bounded multiplicative distortions.

We find it noteworthy that our technique completely captures, with only one lemma, the entire nature of the geometric problem within these constant distortion factors of the pairwise events. However, due to the nature of the incremental selection process, there is a strong dependence between the events in $r(j)$, e.g. between $X_{i,j}$ and $\{X_{i+1,j}, \ldots, X_{j-1,j}\}$. We circumnavigate this obstacle using conditional expectations in the proof of our tail bound.

▶ **Theorem 3.2.** *There is a constant $\lambda > 1$ such that, for every set S of n non-crossing segments, we have $\Pr[\sum_{j=2}^{n}\sum_{i=1}^{j-1} X_{i,j} > \lambda n] < 1/e^n$.*

Proof. For the Chernoff Method, set $t := \ln 2$ and $B := \frac{12\gamma+1}{\ln 2}n$. To leverage Equation (2) and (3) for our events, we regroup the summation terms by column index. Let $C_i := \sum_{Y \in c(i)} Y$ for each $1 \leq i < n$. For $\Pr[\sum_{i<n} C_i > B]$, Markov's inequality gives that

$$\Pr\left[\exp\left(t\sum_{i<n} C_i\right) > e^{tB}\right] \leq \mathbb{E}\left[\exp\left(t\sum_{i<n} C_i\right)\right]/e^{tB}\ , \tag{5}$$

where $\exp(x) = e^x$. Defining $Q_i := \exp(tC_1 + \ldots + tC_i)$, we will show by induction that $\mathbb{E}[Q_{n-1}|s_n] \leq \exp(\mathbb{E}[\sum_{i<n} C_i|s_n])$ for each $s_n \in S$. The conditioning $(s_i, \ldots, s_n)$ denotes that the permutations $\mathbf{P}(S)$ are restricted to those that have this suffix of elements.

For $i = 1$ and each suffix condition $(s_2, \ldots, s_n)$, we have

$$\begin{aligned}
&\mathbb{E}[Q_1|s_2, \ldots, s_n] = \mathbb{E}[\prod_{j=2}^{n} e^{tX_{1,j}}|s_2, \ldots, s_n] \\
=&\prod_{j=2}^{n} \mathbb{E}[e^{tX_{1,j}}|s_2, \ldots, s_n] \\
=&\prod_{j=2}^{n} \Big((1 - \mathbb{E}[X_{1,j}|s_2, \ldots, s_n])e^0 + \mathbb{E}[X_{1,j}|s_2, \ldots, s_n]e^t\Big) \\
=&\prod_{j=2}^{n} \Big(1 + \underbrace{(e^t - 1)}_{=1}\mathbb{E}[X_{1,j}|s_2, \ldots, s_n]\Big) \\
\leq&\exp\Big(\mathbb{E}[\sum_{j=2}^{n} X_{1,j}|s_2, \ldots, s_n]\Big) = \exp\Big(\mathbb{E}[C_1|s_2, \ldots, s_n]\Big)\ ,
\end{aligned}$$

where the second equality is due to Equation (2) under the given suffix conditioning. The third equality is due to the definition of expected values, the fourth due to the distributive law, and the fifth equality due to our choice of t. The inequality is due to $1 + x \leq e^x$.

For $i > 1$ and each condition $(s_{i+1}, \ldots, s_n)$, let $S' = S \setminus \{s_{i+1}, \ldots, s_n\}$ and we have

$$\begin{aligned}
&\mathbb{E}\left[Q_i\middle|s_{i+1}, \ldots, s_n\right] \\
=&\frac{1}{i}\sum_{s_i \in S'} \mathbb{E}\left[Q_{i-1} \cdot e^{tC_i}\middle|s_i, s_{i+1}, \ldots, s_n\right] \\
=&\frac{1}{i}\sum_{s_i \in S'} \mathbb{E}\Big[\underbrace{\mathbb{E}[Q_{i-1}|c(i), s_i, \ldots, s_n]}_{=\mathbb{E}[Q_{i-1}|s_i, \ldots, s_n]} e^{tC_i}\Big|s_i, \ldots, s_n\Big] \\
\leq&\frac{1}{i}\sum_{s_i \in S'} \exp(\mathbb{E}[C_1 + \ldots + C_{i-1}|s_i, \ldots, s_n] \cdot \underbrace{\mathbb{E}[e^{tC_i}|s_i, \ldots, s_n]}_{\leq \exp(\mathbb{E}[C_i|s_i, \ldots, s_n])}) \\
\leq&\frac{1}{i}\sum_{s_i \in S'} \exp(\mathbb{E}[C_1 + \ldots + C_i|s_i, \ldots, s_n] \\
=&\exp(\mathbb{E}[C_1 + \ldots + C_i|s_{i+1}, \ldots, s_n]).
\end{aligned}$$

The first equality is due to that every element of S' is equally likely to be picked for s_i. The second equality is due to the 'law of total expectation'. The third equality is due to a property of our events, see Equation (1). The resulting terms are bounded by the induction hypothesis and analogously to the case $i = 1$, but using Equation (3) for the events $c(i)$ instead. This concludes the induction.

Since $\mathbb{E}[Q_{n-1}] = \frac{1}{n}\sum_{s_n \in S} \mathbb{E}[Q_{n-1}|s_n]$, we have that $\mathbb{E}[Q_{n-1}] \leq \exp\left(\mathbb{E}[\sum_{i<n} C_i]\right)$. Now we change the summation order back to rows first to use $\mathbb{E}[\sum_{j\geq 2}\sum_{i<j} X_{i,j}] \leq 12\gamma n$ from Lemma 3.1.

Hence (5) gives an exponentially decaying tail bound of e^{-n}. ◀

This complements the known high probability bound for the point location cost[2] and shows that the TSD has, with *very high probability*, size $\mathcal{O}(j)$ after every insertion step j. Since the RIC time for the TSD solely entails point location costs and search node creations, we have shown the following statements.

▶ **Corollary 3.3.** *RIC of a TSD for n non-crossing segments takes w.h.p. $\mathcal{O}(n \ln n)$ time.*

▶ **Corollary 3.4.** *The TSD size, for n non-crossing segments, can be made $\mathcal{O}(n)$ with 'rebuild if too large' by merely increasing the expected construction time by an additive constant.*

Acknowledgments

The authors want to thank Wolfgang Mulzer for pointing out an incorrect statement in an earlier draft, Boris Aronov for discussions during his stay, Daniel Bahrdt for the github project `OsmGraphCreator`, and Raimund Seidel for sharing his excellent lecture notes on a CG course he thought 1991 at UC Berkeley.

References

1 Pankaj K. Agarwal, Lars Arge, Jeff Erickson, Paolo Giulio Franciosa, and Jeffrey Scott Vitter. Efficient searching with linear constraints. In *Proc. of the 17th Symposium on Principles of Database Systems (PODS'98)*, pages 169–178, 1998. doi:10.1145/275487.275506.

2 Pankaj K. Agarwal, Boris Aronov, Sariel Har-Peled, Jeff M. Phillips, Ke Yi, and Wuzhou Zhang. Nearest neighbor searching under uncertainty II. In *Proc. of the 32nd Symposium on Principles of Database Systems (PODS'13)*, pages 115–126, 2013. doi:10.1145/2463664.2465219.

3 D. S. Andrews, J. Snoeyink, J. Boritz, T. Chan, G. Denham, J. Harrison, and C. Zhu. Further comparison of algorithms for geometric intersection problems. In *Proc. 6th International Symposium on Spatial Data Handling*, 1994. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.18.2254.

4 D. S. Andrews and Jack Snoeyink. Geometry in GIS is not combinatorial: Segment intersection for polygon overlay. In *Proc. of 11th Symposium on Computational Geometry (SoCG'95)*, pages C24–C25, 1995. URL: https://doi.org/10.1145/220279.220333, doi:10.1145/220279.220333.

5 Guy E. Blelloch, Yan Gu, Julian Shun, and Yihan Sun. Parallelism in randomized incremental algorithms. *J. ACM*, 67(5):27:1–27:27, 2020. doi:10.1145/3402819.

6 Milutin Brankovic, Nikola Grujic, André van Renssen, and Martin P. Seybold. A simple dynamization of trapezoidal point location in planar subdivisions. In *Proc. 47th International Colloquium on Automata, Languages, and Programming (ICALP'20)*, pages 18:1–18:18, 2020. doi:10.4230/LIPIcs.ICALP.2020.18.

7 Timothy M. Chan. A simple trapezoid sweep algorithm for reporting red/blue segment intersections. In *Proc. of the 6th Canadian Conference on Computational Geometry (CCCG'94)*, pages 263–268, 1994.

[2] Cf. [9, Chapter 6.4] and [17, Lemma 3.1.5 and Theorem 3.1.4].

8 Kenneth L. Clarkson, Kurt Mehlhorn, and Raimund Seidel. Four results on randomized incremental constructions. *Computational Geometry: Theory and Applications*, 3:185–212, 1993. doi:10.1016/0925-7721(93)90009-U.

9 Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational Geometry: Algorithms and Applications, 3rd Edition*. Springer, 2008. doi:10.1007/978-3-540-77974-2.

10 Michael Hemmer, Michal Kleinbort, and Dan Halperin. Improved implementation of point location in general two-dimensional subdivisions. In *Proc. 20th European Symposium on Algorithms (ESA'12)*, pages 611–623, 2012. doi:10.1007/978-3-642-33090-2_53.

11 Michael Hemmer, Michal Kleinbort, and Dan Halperin. Optimal randomized incremental construction for guaranteed logarithmic planar point location. *Computational Geometry: Theory and Applications*, 58:110–123, 2016. doi:10.1016/j.comgeo.2016.07.006.

12 Jirí Matoušek and Raimund Seidel. A tail estimate for mulmuley's segment intersection algorithm. In *Proc. 19th International Colloquium on Automata, Languages and Programming (ICALP'92)*, pages 427–438, 1992. doi:10.1007/3-540-55719-9_94.

13 Kurt Mehlhorn, Micha Sharir, and Emo Welzl. Tail estimates for the space complexity of randomized incremental algorithms. In *Proc. of the 3rd Symposium on Discrete Algorithms (SODA'93)*, pages 89–93, 1992. URL: http://dl.acm.org/citation.cfm?id=139404.139423.

14 Ketan Mulmuley. A fast planar partition algorithm, I. *J. of Symbolic Computation*, 10(3-4):253–280, 1990. doi:10.1016/S0747-7171(08)80064-8.

15 Ketan Mulmuley. A fast planar partition algorithm, II. *J. ACM*, 38(1):74–103, 1991. doi:10.1145/102782.102785.

16 Ketan Mulmuley. Randomized multidimensional search trees: Lazy balancing and dynamic shuffling. In *Proc. of the 32nd Symposium on Foundations of Computer Science (FOCS'91)*, pages 180–196, 1991. doi:10.1109/SFCS.1991.185368.

17 Ketan Mulmuley. *Computational Geometry: An Introduction Through Randomized Algorithms*. Prentice Hall, 1994.

18 Raimund Seidel. A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. *Computational Geometry: Theory and Applications*, 1:51–64, 1991. doi:10.1016/0925-7721(91)90012-4.

19 Raimund Seidel. *Backwards Analysis of Randomized Geometric Algorithms*, pages 37–67. Springer Berlin Heidelberg, Berlin, Heidelberg, 1993. doi:10.1007/978-3-642-58043-7_3.

20 Raimund Seidel. Teaching computational geometry. In *Proc. of the 5th Canadian Conference on Computational Geometry (CCCG'93)*, pages 272–272, 1993.

21 Raimund Seidel and Cecilia R. Aragon. Randomized search trees. *Algorithmica*, 16(4-5):464–497, 1996. doi:10.1007/BF01940876.

22 Sandeep Sen. A unified approach to tail estimates for randomized incremental construction. In *Proc. of the 36th Symposium on Theoretical Aspects of Computer Science (STACS'19)*, pages 58:1–58:16, 2019. doi:10.4230/LIPIcs.STACS.2019.58.

23 Jean Vuillemin. A unifying look at data structures. *Commun. ACM*, 23(4):229–239, 1980. URL: https://doi.org/10.1145/358841.358852, doi:10.1145/358841.358852.

24 Ron Wein, Eric Berberich, Efi Fogel, Dan Halperin, Michael Hemmer, Oren Salzman, and Baruch Zukerman. 2D arrangements. In *CGAL User and Reference Manual*. 2020. URL: https://doc.cgal.org/5.1.1/Manual/packages.html#PkgArrangementOnSurface2.

Shortest Paths in Portalgons

Maarten Löffler[1], Rodrigo I. Silveira[2], and Frank Staals[1]

1 **Department of Information and Computing Sciences, Utrecht University, Netherlands**
`m.loffler@uu.nl, f.staals@uu.nl`

2 **Department de Matemàtiques, Universitat Politècnica de Catalunya, Spain**
`rodrigo.silveira@upc.edu`

1 Introduction

We introduce *portalgons*, collections of polygons with some edges identified, as an abstract generalization of simple polygons, polygons with holes, polyhedral surfaces, and even ruled surfaces. We are interested in understanding shortest paths in portalgons. To the best of our knowledge, this is a rather unexplored concept which, though it has been long adopted into popular culture [3, 5, 11], has only been studied from a computational point of view in the context of annular ray shooting by Erickson and Nayyeri [6]. The concept of portalgons is related to the topological representation of surfaces by polygons with edges identified (see, e.g., [9, Chapter 6]). Shortest paths have been studied in many different geometric settings, such as simple polygons, polygonal domains, terrains, surfaces, and polyhedra (see, e.g., [1, 2, 7, 8]; refer to [10] for a comprehensive survey).

Portalgons. We define a *portalgon* $\mathcal{P}$ to be a tuple $(\mathcal{F}, P)$, where $\mathcal{F}$ is a collection of polygons, called *fragments*, and P is a collection of portals. We assume that all fragments in $\mathcal{F}$ are simple polygons.[1] A *portal* is a pair $(e, \hat{e})$ of directed, equal length, edges from some fragments (possibly the same) of $\mathcal{F}$. We refer to e and $\hat{e}$ as *portal edges*. In particular, $\hat{e}$ is the *twin* of e, and vice versa. See Fig. 1. If p is a point on a portal edge e, then $\hat{p}$ will denote the corresponding point on $\hat{e}$. More precisely, let $e = \overrightarrow{uv}$ and $\hat{e} = \overrightarrow{wz}$ be a portal, where u, v are vertices of the fragment containing e and w, z are vertices of the fragments containing $\hat{e}$. If $p = \lambda u + (1-\lambda)v$, for some $\lambda \in (0, 1)$, then $\hat{p} = \lambda w + (1-\lambda)z$.[2]

Let n be the total number of vertices in (the fragments of) $\mathcal{P}$, and let m be the number of portal edges. We will require that no edge is part of multiple portals, and thus the number

[1] It will be clear later that this does not restrict the problem, since any non-simple fragment can be represented by a collection of simple fragments by using additional portals.

[2] If a vertex is incident to two portal edges then the vertex twin might not be unique—in this case a set of points of any cardinality might be identified.

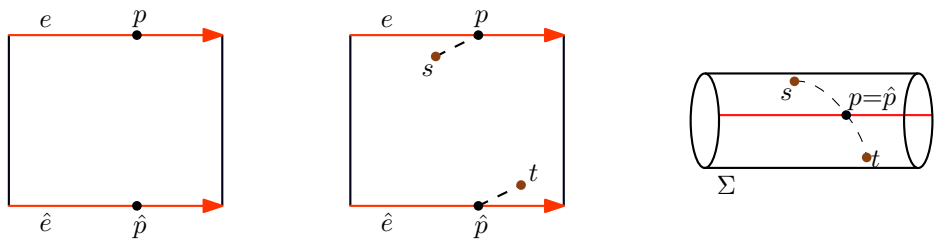


Figure 1 Left: A portalgon consisting of one fragment with one portal (and two portal edges, e and $\hat{e}$). Points p and $\hat{p}$ are identified as twins. Middle: the shortest path from s to t (dashed) goes through the portal. Right: the surface Σ associated to this portalgon is a cylinder.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

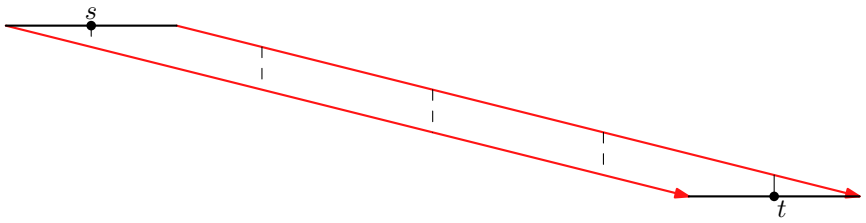


Figure 2 The complexity of the shortest path between s and t is independent of n and m.

of portals is $m/2$. Furthermore note that $m \leq n$. We denote the number of vertices and the number of portal edges in a fragment $F \in \mathcal{F}$ by n_F and m_F, respectively.

We assume that for any portal $(e, \hat{e})$ the portal edges have opposite orientations, i.e., the interior of the fragment containing e lies to the right of e and the interior of the fragment containing $\hat{e}$ lies to the left of $\hat{e}$. As a result, the portalgon $\mathcal{P}$ describes an orientable surface (2-manifold with boundary) Σ, see Fig. 1(right). Specifically, Σ is the space obtained from $\mathcal{P}$ by taking the collection $\mathcal{F}$ and identifying corresponding pairs of points on portal edges (i.e., p with $\hat{p}$, for each p in a portal edge). Note that Σ does not necessarily have a non-intersecting embedding in 3-space. Any point p in Σ maps to a point or a set of identified points in $\mathcal{P}$ (the latter happens when p maps to a point on a portal edge). With some abuse of notation we may therefore write $\mathcal{P}$ to mean Σ when it causes no confusion.

Shortest paths. Let $\pi(p, q)$ denote a geodesic *shortest path* on Σ connecting p to q. The length of this path is the (geodesic) distance between p and q. A path $\pi(p, q)$ uniquely corresponds to an alternating sequence $p = v_0, s_0, v_1, s_1, .., s_k, v_{k+1} = q$ of points and maximal open-ended segments. In particular, any point v_i, with $i \in 1, .., k$, is either a portalgon vertex or a point on a portal edge, and any segment s_i is a straight line segment that connects two such points v_i and v_{i+1} and is completely contained in a single fragment of $\mathcal{P}$. We define the *complexity* of the path as the length of this sequence, i.e., $2k + 3$.[3]

In this work we are interested in understanding shortest paths in portalgons. As seen in the simple example of Fig. 1, a shortest path can go through one or multiple portals. More interestingly, it can go through the same portal multiple times, as shown in Fig 2. In that example, the complexity of $\pi(s, t)$ can be increased arbitrarily by moving the bottom segment horizontally towards the right. This shows that the complexity of a shortest path may not be bounded by any function of n and m, in which case we will say that it is *unbounded*. This fact was already observed in [6], where it is shown that such a path can be computed in $O(n + \log k)$ time, where k is the number of times the shortest path goes through the portal.

In this paper we would like to understand the conditions that can make a portalgon have high complexity shortest paths, and how to prevent it. It is interesting to note that the increase in the complexity of $\pi(s, t)$ in the example above requires to make the fragment thin and very wide. This may suggest that bounding the aspect ratio of fragments may be enough to obtain constant-complexity paths. However, that is not the case. In the example in Fig. 3, with three fragments and three portals, $\pi(s, t)$ can be made to cross the triangular

[3] We use this, arguably somewhat complicated, definition of complexity of a path to avoid counting "singleton points"; when a path $\pi(p, q)$ passes through a portal then intuitively this portal splits $\pi(p, q)$ into two parts, one part ending at a point v, and the other part starting at point $\hat{v}$. However, if v is an endpoint of the portal, there may be multiple copies of v, all of which lie on $\pi(p, q)$. However, $\pi(p, q)$ "continues" only through one these endpoints. Hence, we want to avoid counting these other points as part of the complexity of $\pi(p, q)$.

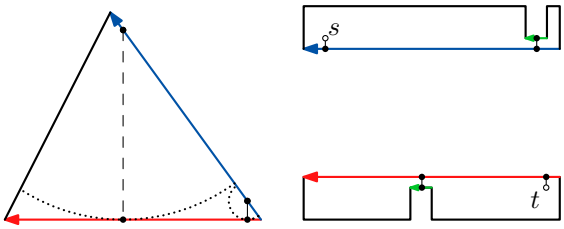


Figure 3 The shortest path from s to t can go through the left (triangular) fragment an unbounded number of times, even though the fragment is *fat*. (The depicted path is shown schematically.)

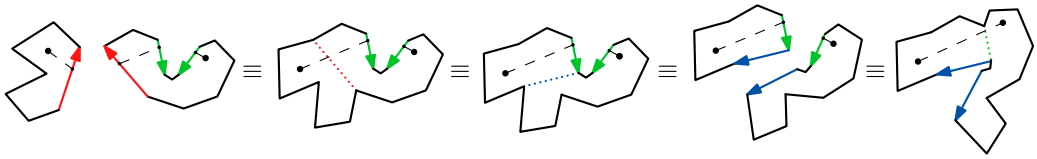


Figure 4 Five equivalent portalgons, with the shortest path between (the same) two points.

fragment as many times as desired—i.e., a number of times linear in n—even though that fragment is *fat* [4]. To that end, the distance to the green "helper" portal has to be made shorter than depicted, arbitrarily close to 0. Then the shortest path between s and t would be as in the figure, crossing the fat fragment from blue to red twice. We can repeat the construction in the figure, adding more helper portals, but each time we have to zoom in, leading to exponential scale.

Contributions. In this work we study shortest paths in portalgons to understand when they can be unbounded, and how that can be avoided. We will show that if a portalgon with only one portal contains an unbounded shortest path, there is always an *equivalent* portalgon with the property that all shortest paths have a complexity which can be expressed as a function of n and m. We also study the computational problem of finding such an equivalent portalgon: under some conditions (when the portal edges are parallel and we are allowed the use of the floor operation) this can be done in linear time, but in general, the time required may be unbounded in n and m.

2 Portalgons and happiness

In this section we formalize several concepts that will be useful to understand portalgons.

Equivalent portalgons. Given a portalgon $\mathcal{P}$, there are many other portalgons that describe the same underlying space Σ. For instance, we can always cut a fragment into two smaller fragments by transforming a chord of the fragment into a portal, or, assuming this does not cause any overlap, we can glue two fragments along a portal, see Fig. 4. More formally, two portalgons $\mathcal{P}$ and $\mathcal{Q}$ are *equivalent*, denoted $\mathcal{P} \equiv \mathcal{Q}$, when for any pair of points $s, t \in \mathcal{P}$ the distance between them is the same in both, or in other words, when there is an isometry between them.

Happiness. Our ultimate goal will be to find, for a given input portalgon, an equivalent portalgon such that all its shortest paths have bounded complexity. To this end, we introduce the notion of a *happy portalgon*, and more specifically a *happy fragment* of a portalgon.

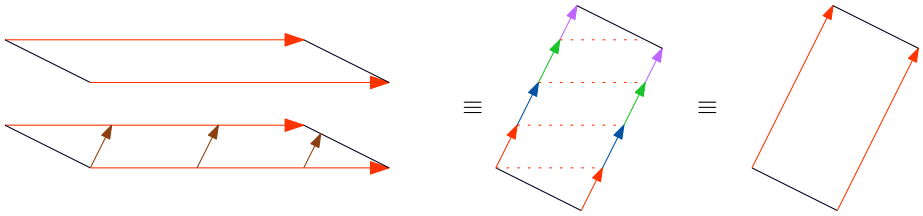


Figure 5 Left: a fragment with two parallel edges and $\Delta \neq 0$. Center: result of cutting the fragment along a perpendicular ray, resulting in a new fragment with several portals, which is equivalent to a fragment with one portal and $\Delta = 0$ (right).

Let $\mathcal{P} = (\mathcal{F}, P)$ be a portalgon, let χ be a path in $\mathcal{P}$, and let $F \in \mathcal{F}$ be a fragment in $\mathcal{F}$. We define $c(X)$ as the number of connected components in X. The *happiness* $\mathcal{H}(F)$ of fragment F is defined as $\mathcal{H}(F) = \max_{p,q \in \mathcal{P}} c(F \cap \pi(p,q))$. The happiness of $\mathcal{P}$ is then defined as $\mathcal{H}(\mathcal{P}) = \max_{F \in \mathcal{F}} \mathcal{H}(F)$. We say that a portalgon is h-happy when $\mathcal{H}(\mathcal{P}) \leq h$.

Note that in an h-happy portalgon, a shortest path crosses every portal at most h times. Therefore, if $\mathcal{P}$ is an h-happy portalgon with n vertices and m portals, the shortest path between any two points s and t in $\mathcal{P}$, $\pi(s,t)$, has complexity $O(n + hm)$.

The remainder of this work is devoted to exploring whether any portalgon can be transformed into an equivalent one that is $O(1)$-happy, something that we believe to be true.

▶ Conjecture (Universal Happiness). Let $\mathcal{P}$ be a portalgon with n vertices. There exists a portalgon $\mathcal{P}' \equiv \mathcal{P}$ with $O(n)$ vertices that is $O(1)$-happy.

3 Making portalgons happy

In this section we study how to rearrange a portalgon into an equivalent happy portalgon. Since any fragment with only one portal edge is happy, we sketch our main ideas for the next simplest case: a portalgon consisting of a single fragment F with exactly two *parallel* portal edges. We assume w.l.o.g. that the portal edges e and $\hat{e}$ are horizontal, e is above $\hat{e}$ and e starts further to the left. Let Δ and v be the horizontal, resp. vertical, distance between the start vertices of e and $\hat{e}$.

▶ **Lemma 1.** *If $\Delta = 0$, then the fragment is 2-happy.*

If $\Delta \neq 0$, the fragment might be happy or not. Next we present a method to make a fragment happy by splitting it into smaller fragments. Let $z = \Delta / v$. Conceptually, from the start vertex of $\hat{e}$ we shoot a ray with slope z in the interior of F, until we hit a boundary (i.e., non-portal) edge. Then we cut along this ray, which results in several smaller fragments, which we finally glue together again. Refer to Figure 5. This leads to the following result.

▶ **Lemma 2.** *Let F be a parallelogram with two parallel portal edges. There is always an equivalent parallelogram F' with $\Delta = 0$. This parallelogram F' is 2-happy.*

If F is not a parallelogram, it may happen that when gluing together the new smaller fragments we obtain a non-simple polygon. Therefore we cannot always transform F into a single equivalent happy fragment; refer to Figure 6. Fortunately, we can still transform any fragment into a constant number of happy fragments; that is, we can prove the Universal Happiness Conjecture for the case of one fragment with two parallel portal edges. Refer to Figure 7.

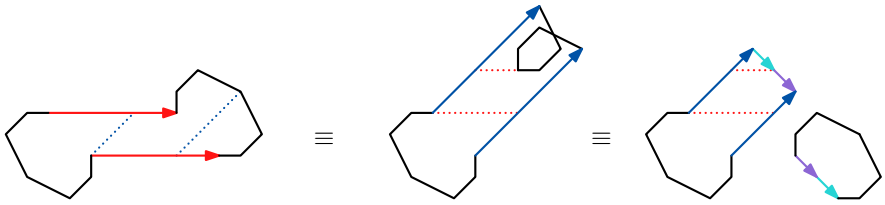


Figure 6 Example of fragment (left) where the cutting along the perpendicular ray produces a non-simple polygon (center). However, here this can be fixed using one more fragment (right).

▶ **Lemma 3.** *Consider a fragment F with two horizontal portals e and $\hat{e}$. Let m be the line through the start points of e and $\hat{e}$. If F is not 5-happy, then there is a line segment parallel to m from a point on e to a point on $\hat{e}$ that lies completely within F.*

▶ Observation 4. Let F be a simple fragment, let $\overline{ab}$ be a line segment inside F. A single connected component $\pi \cap F$ of a shortest path π can intersect $\overline{ab}$ in at most one component.

Observation 4 implies that the two endpoints p and q of a maximal component $\pi(p, q)$ of a shortest path in F cannot lie on the same portal-edge e unless $\pi(p, q) = \overline{pq} \subseteq e$.

▶ **Theorem 5.** *Let $\mathcal{P}$ be a portalgon with one fragment F with n vertices, and one portal whose edges are parallel. There exists a 5-happy portalgon $\mathcal{P}'$ equivalent to $\mathcal{P}$ consisting of at most three fragments and total complexity $O(n)$.*

Proof. Assume without loss of generality that both portal edges e and $\hat{e}$ are horizontal and oriented left-right. We now argue that when no three vertices of F are colinear, and F is not already 5-happy, we can split F into at most seven 4-happy fragments of total complexity $O(n)$. Refer to Figure 7. Finally, we show how to reduce the number of fragments to three, while remaining 4-happy, even without the general position assumption.

Let m be the line through the start points of e and $\hat{e}$. By Lemma 3, if there is no translate of m whose intersection with F contains a segment connecting e to $\hat{e}$, F is already 5-happy. Let m_ℓ be the leftmost such translate of m and m_r the rightmost such translate; m_ℓ contains a vertex ℓ of F and m_r contains a vertex r of F (possibly, ℓ or m is an endpoint of e or $\hat{e}$). Let a and $\hat{a}$ be the intersection points of m_ℓ with e and $\hat{e}$, and let b and $\hat{b}$ be the intersection points of m_r with e and $\hat{e}$. We cut the parallelogram $Z = ab\hat{a}\hat{b}$ from F, which splits F into at most seven fragments (since, by general position, $\overline{a\hat{a}}$ and $\overline{b\hat{b}}$ contain at most two reflex vertices each). Using a transformation similar to that of Lemma 2, we turn Z into a 4-happy fragment (the fact that the sides $\overline{a\hat{a}}$ and $\overline{b\hat{b}}$ are actually portal edges increases the happiness by at most two). Let T and B be the fragments containing the starting points of e and $\hat{e}$, respectively. We argue that T is 4-happy. The argument that B is 4-happy is symmetric. The same holds for the fragments containing the endpoints of e and $\hat{e}$. Any other fragments (if they exist) contain only one portal edge and are thus already 2-happy.

Consider the connected components of a shortest path $\pi = \pi(s, t)$ with F. By Observation 4 such a component either: (i) contains s, (ii) contains t, or (iii) connects a point p_i on e to a point q_i on $\hat{e}$. Again by Observation 4 each such component can intersect $\overline{a\hat{a}}$ at most once, so each such component can intersect T at most once.

We now further distinguish the type (iii) components into three types, depending on whether p_i lies on the part of e in T and whether q_i lies on the part of $\hat{e}$ in B. If p_i lies outside T the component does not intersect T at all (by Observation 4). There is only one component for which p_i lies inside T, and q_i lies inside B that can intersect T (as such a component must go through point ℓ), and one component for which p_i lies in T and q_i lies

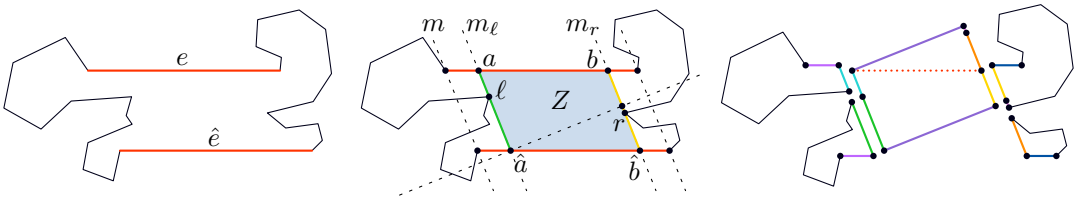


Figure 7 (a) Fragment F with two parallel portals (b) The lines m_ℓ and m_r intersecting e and $\hat{e}$ in $a, \hat{a}$ define a parallelogram $Z = ab\hat{a}\hat{b}$ that splits F into at most seven sub fragments. (c) The resulting set of 5-happy fragments. Note that it is possible to reduce the number of fragments by shifting m_ℓ slightly to the right and m_r slightly to the left.

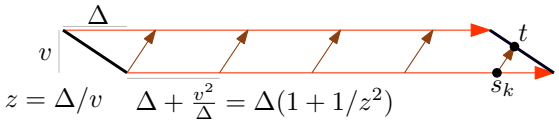


Figure 8 Finding the last intersection of the ray with the portal.

outside of B. The main idea here is that $\pi(p_i, q_i)$ "blocks" the remainder of π from entering B again, and thus p_i is the last component of type (iii) that intersects T.

It follows that there are thus only four components of $\pi \cap F$ that intersect T, and each such component intersects T in only one consecutive subpath. Hence T is 4-happy.

We conclude that the portalgon $\mathcal{P}$' that we end up with is 4-happy, equivalent to $\mathcal{P}$, and has at most $|\mathcal{F}| + 6$ fragments. Furthermore, every vertex of $\mathcal{P}$ appears in at most $O(1)$ fragments of $\mathcal{P}'$, and thus $\mathcal{P}'$ has complexity $O(n)$.

Finally, observe that (before splitting F) we can actually shift the left and right sides of Z inwards by some arbitrarily small ε. It then follows that F is now split into only three fragments, two of which are already 4-happy. We transform the remaining fragment (parallelogram Z) into a 2-happy parallelogram as before. We now get only three fragments (of total complexity $O(n)$), even if F contains three or more colinear vertices. ◀

Let k be the number of times the ray hits the portal. Since k is unbounded in terms of n, computing a happy portalgon equivalent to $\mathcal{P}$ is non-trivial. We can compute such a portalgon in $O(n + \log(1 + k))$ time, or in $O(n)$ time if we can use the *floor* operation. The essential part of the computation is finding the last time a ray intersects the portal; refer to Figure 8. Explicitly computing all intersection points would lead to an additive $O(k)$ term. Instead, we can perform exponential search by noting that the horizontal distance between each pair of consecutive intersection points is the same.

4 Beyond Parallel Portals

In the full version, we also prove the Universal Happiness Conjecture for portalgons that consist of one fragment with two non-parallel portal edges; see Figure 9. Computing an equivalent happy portalgon in this case is more involved; using ideas from [6] we may still obtain the same running time. When portalgons have more than one fragment, we may consider the *fragment graph* (Figure 10), which is a multigraph with one vertex per fragment and an edge for every portal. For some fragment graphs we can also prove the conjecture. For instance, if the fragment graph is acyclic, the portalgon is already happy. If it is almost acyclic but has a single fragment with a single self-loop, we can still apply the techniques

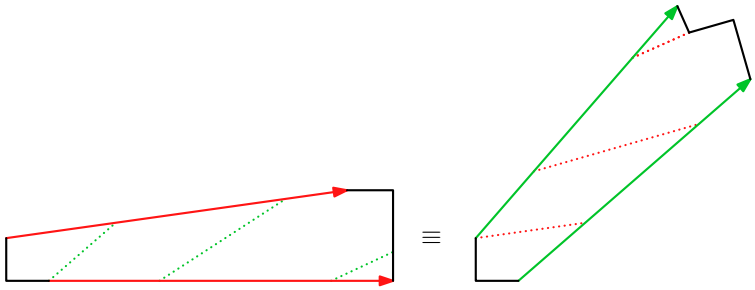


Figure 9 A fragment with non-parallel portal edges, and an equivalent happy one.

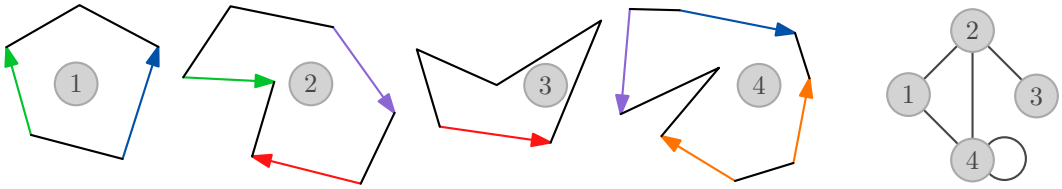


Figure 10 The fragment graph of a portalgon.

described above. However, we do not know how to approach a proof of the Universal Happiness Conjecture in general. Even the behavior of shortest paths in a single fragment with two self-loops is non-trivial; see for instance Figure 11.

References

1 Jindong Chen and Yijie Han. Shortest paths on a polyhedron. In *Proceedings of the Sixth Annual Symposium on Computational Geometry*, SCG '90, page 360–369, New York, NY, USA, 1990. Association for Computing Machinery. URL: https://doi.org/10.1145/98524.98601, doi:10.1145/98524.98601.

2 Jindong Chen and Yijie Han. Shortest paths on a polyhedron. *Int. J. Comput. Geom. Appl.*, 6(2):127–144, 1996. URL: https://doi.org/10.1142/S0218195996000095, doi:10.1142/S0218195996000095.

3 Valve Corporation. Portal, 2007. Video game.

4 Alon Efrat. The complexity of the union of (alpha, beta)-covered objects. *SIAM J. Comput.*, 34(4):775–787, 2005. URL: https://doi.org/10.1137/S0097539702407515, doi:10.1137/S0097539702407515.

5 H Ellison. The city on the edge of forever, 1967. Star Trek, season 1, episode 28.

6 Jeff Erickson and Amir Nayyeri. Tracing compressed curves in triangulated surfaces. *Discret. Comput. Geom.*, 49(4):823–863, 2013. URL: https://doi.org/10.1007/s00454-013-9515-z, doi:10.1007/s00454-013-9515-z.

7 Leonidas J. Guibas and John Hershberger. Optimal shortest path queries in a simple polygon. *Journal of Computer and System Sciences*, 39(2):126 – 152, 1989. URL: http://www.sciencedirect.com/science/article/pii/002200008990041X, doi:https://doi.org/10.1016/0022-0000(89)90041-X.

Figure 11 Even in a portalgon with only one fragment and two portals, shortest paths may have nontrivial structure. The example passes through blue-blue-blue-red-blue-blue-blue-red-blue-blue.

8 John Hershberger and Subhash Suri. An Optimal Algorithm for Euclidean Shortest Paths in the Plane. *SIAM Journal on Computing*, 28(6):2215–2256, 1999.

9 J. Lee. *Introduction to Topological Manifolds*. Graduate Texts in Mathematics. Springer New York, 2nd edition, 2010.

10 Joseph S. B. Mitchell. Shortest paths and networks. In Jacob E. Goodman, Joseph O'Rourke, and Csaba D. Toth, editors, *Handbook of Discrete and Computational Geometry, Third Edition*, pages 811–848. Chapman and Hall/CRC, 2017.

11 A. Wachowski and L. Wachowski. Matrix revolutions, 2003. Warner Bros. Motion picture.

Coordinating Programmable Matter via Shortest Path Trees

Irina Kostitsyna[1], Tom Peters[1], and Bettina Speckmann[1]

1 **TU Eindhoven, the Netherlands**
[i.kostitsyna|t.peters1|b.speckmann]@tue.nl

1 Introduction

Programmable matter is a smart material composed of a large quantity of robot particles capable of communicating locally, performing simple computation, and, based on the outcome of this computation, changing their physical properties. Particles are able to move through a programmable matter system by changing their geometry and attaching to (and detaching from) neighboring particles. Thus, a particle system can be programmed to reconfigure its global shape by changing local adjacencies between particles. Shape assembly and reconfiguration of particle systems have attracted a lot of interest in the algorithmic community in the past decade and a variety of specific models have been proposed [1, 2, 8, 10, 12, 14, 16]. We focus on the Amoebot model [7], which we briefly introduce below, refer to Daymude et al. [5] for additional details. The Amoebot model is highly distributed and hence a number of algorithmic primitives have been developed to coordinate actions of particles [4, 9, 15].

The Amoebot model. Particles occupy nodes of a triangular grid G embedded in the plane. A particle can occupy one (contracted particle) or two (expanded particle) adjacent nodes of the grid, and can communicate with its neighboring particles (see Figure 1 (left)). The particles have limited computational power due to constant memory space, they have no common notion of orientation (disoriented), and no common notion of clockwise (cw) or counter-clockwise (ccw) order (no consensus on chirality). They are identical (no IDs and they all execute the same algorithm), but can locally distinguish between their neighbors using six (for contracted particles) or ten (for expanded particles) *port identifiers*. Ports are labeled in order (either cw or ccw) modulo six or ten, respectively. Particles communicate by writing into a register of their neighbors using the ports.

Particles can move in two different ways: a contracted particle can *expand* into an adjacent empty node of the grid, and an expanded particle can *contract* into one of the nodes it currently occupies. Each node of G can be occupied by at most one particle, and we require that the particle system stays connected at all times. To preserve connectivity more easily, we allow a *handover* variant of both move types, a simultaneous expansion and contraction of two neighboring particles using the same node (see Figure 1 (right)). The handover move

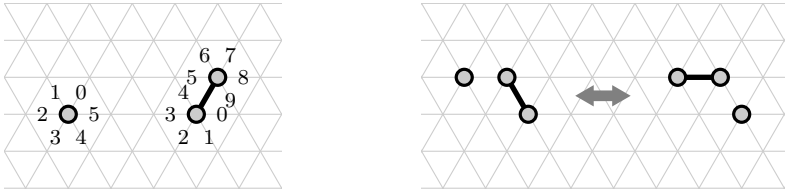


Figure 1 Left: particles with ports labeled, in contracted and expanded state. Right: handover operation for preserving system connectivity. A particle can expand while pushing an expanded neighbor, and an expanded particle can contract while pulling a contracted neighbor.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

can be initiated by any of the two particles; if it is initiated by the expanded particle, we say it *pulls* its contracted neighbor, otherwise we say that it *pushes* its expanded neighbor.

Particles operate in activation cycles: when activated, they can read from the memory of their immediate neighbors, compute, write into the memory of their neighbors, and perform a move operation. For simplicity of presentation, we assume that particles are activated one by one in an order given by an adversarial but fair scheduler (at any moment in time t, for any particle, it must be activated at some time in the future $t' > t$). Analysis of algorithms for such sequential scheduler can be easily extended to arbitrary asynchronous activation of particles under some assumptions [5]. We perform running time analysis in terms of the number of *rounds*, time intervals in which all particles have been activated at least once.

We call the set of particles and their internal states a *particle configuration* $\mathcal{P}$. Let $G_{\mathcal{P}}$ be the subgrid of G induced by the nodes occupied by particles in $\mathcal{P}$. We say that $\mathcal{P}$ is *connected* if there is a path in $G_{\mathcal{P}}$ between any two particles in $\mathcal{P}$. A *hole* in $\mathcal{P}$ is an interior face of $G_{\mathcal{P}}$ with more than three vertices. A particle configuration $\mathcal{P}$ is *simply connected* if it is connected and has no holes. We say that a particle $q \in \mathcal{P}$ is $\mathcal{P}$*-visible* from a particle $p \in \mathcal{P}$ if there exists a shortest path from p to q in G that is contained in $G_{\mathcal{P}}$. This definition of visibility is closely related to staircase visibility (or s-visibility) in rectilinear polygons [3, 11]. Finally, we say that $\mathcal{P}$ is *convex* if for every pair of particles $p, q \in \mathcal{P}$, p is $\mathcal{P}$-visible from q.

Contribution We propose a new primitive for programmable matter, a *shortest path tree* (SP-tree), to facilitate and optimize shape reconfiguration in the Amoebot model. Furthermore, we present an efficient algorithm for constructing an SP-tree in a simply connected particle system, using a version of shortest path maps [13] on the grid.

2 Shortest path trees

Among the previously proposed primitives for Amoebot coordination is the *spanning forest primitive* [9] which organizes all particles in a connected component into a tree to facilitate movement of all particles while staying connected. The root initiates the movement, all other particles follow via handovers between parents and children. This primitive does not impose any additional structure on the resulting spanning tree. To facilitate effective movement we propose to construct shortest path trees, which are special spanning trees where the shortest path from a particle to the root within the tree equals the unrestricted shortest path in $G_{\mathcal{P}}$.

To ensure that all paths in the tree are shortest, we need to control the growth of the tree. One way to do so, is to use breadth-first search together with a token passing scheme, which ensures synchronization between growing layers of the tree (see Figure 2). Here a

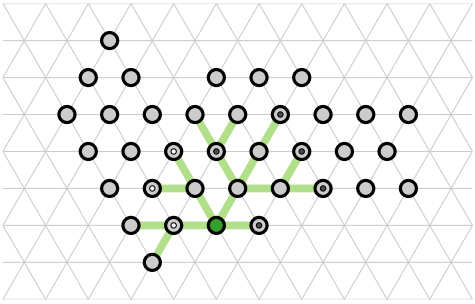


Figure 2 A step in the breadth first search.

token is a constant amount of information which is written into the memory of each particle, possibly by its neighbors, to propagate information. The root (green) sends a *growth token* (white) to all branches; these tokens propagate to the current leaves of the tree. If a leaf p receives a token, then it includes all neighboring particles, which are not part of the tree yet, into the tree with p as their parent, by writing into their memory. Afterwards the leaf creates a *finished growth token* (dark gray) and sends it back to its parent. Once a particle receives finished growth tokens from all of its children it passes one of them up the tree and discards the others. Once the root receives a finished growth token from all its children, it initiates the growth of the next layer by sending a new growth token to all its children. The process terminates once no leaf can grow any further; this information can be encoded in the finished growth token. Breadth first search takes $\mathcal{O}(n^2)$ rounds to complete the tree for a connected particle configuration $\mathcal{P}$ with n particles, since every growth step requires a token to be passed along the complete depth of the tree.

▶ **Lemma 1.** *Given a connected particle configuration $\mathcal{P}$ with n particles, we can create an SP-tree using at most $\mathcal{O}(n^2)$ rounds.*

To create SP-trees efficiently for simply connected particle systems, we describe a version of the shortest path map (SPM) [13] on the grid. Let $\mathcal{P}$ be simply connected, and let $R_0 \subseteq \mathcal{P}$ be the subconfiguration of all particles $\mathcal{P}$-visible from some root particle r. By analogy with the geometric SPM, we refer to R_0 as a *region*. If $R_0 = \mathcal{P}$ then SPM(r) is simply R_0. Otherwise, consider the connected components $\{\mathcal{R}_1, \mathcal{R}_2, \dots\}$ of $\mathcal{P} \setminus R_0$. A *window* of $\mathcal{R}_i$ is a line of particles in R_0 each of which is adjacent to one or more particles in $\mathcal{R}_i$. Let r_i be the closest particle to r of the window W_i of $\mathcal{R}_i$. Then SPM(r) is the union of R_0 and the shortest path maps of all r_i in $\mathcal{R}_i \cup W_i$ for all i. The set of particles $R_i \subseteq \mathcal{R}_i \cup W_i$ is the *visibility region* of r_i, and r_i is the *root* of the region R_i (see Figure 3). Note that by our definition the particles of a window between two adjacent regions of a shortest path map belong to both regions. Observe that any shortest path from a particle $q \in W_i$ to r passes through r_i, and thus any window forms a subpath of some path from r to some leaf in any SP-tree.

▶ **Lemma 2.** *Let r_i be the root of a visibility region R_i. For every particle p in R_i, the shortest path from r to p passes through r_i.*

Proof. Assume that there exists a shortest path π in $G_{\mathcal{P}}$ from r to p that does not pass through r_i. Assume further that R_i is adjacent to R_0 (see Figure 4). Path π must cross window W_i (r_i, w) at some particle $q \neq r_i$. The extension of window (r_i, w), (r_i, w'), partitions R_0 into two parts. Since R_0 is a visibility region, π must cross (r_i, w') at some particle q'.

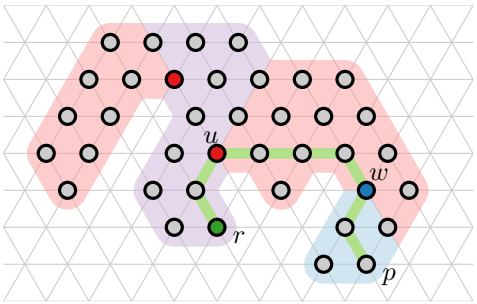


Figure 3 Shortest path map of node r. Any shortest path between r and p must pass through nodes u and w. The region R_0 (in purple) consists of the particles $\mathcal{P}$-visible to r. The two red and the blue particle are the roots of the corresponding SPM regions.

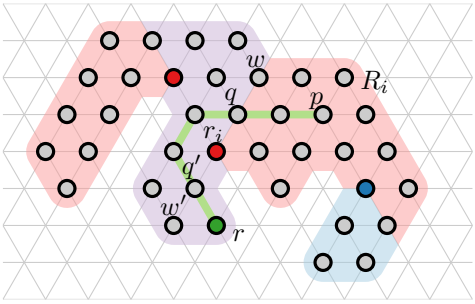


Figure 4 A shortest path from r to a particle in R_i.

Now there exists a shorter path from r to p: from r to q' to q to p. Contradiction. The same argument applies recursively to regions R_i further removed from R_0. ◀

▶ **Corollary 3.** *Any shortest path π between r and any other particle p in $\mathcal{P}$ must pass through the roots of the SPM regions that π crosses.*

If a particle p is $\mathcal{P}$-visible from r then there is a 60°-*angle monotone path* from r to p in $G_{\mathcal{P}}$. That is, there exists a 60°-cone in a fixed orientation, such that for each particle q along the path the remainder of the path lies completely inside this cone translated to q (see Figure 5). Although not named explicitly, angle monotone paths were introduced in [6].

We use a version of such cones to grow an SP-tree efficiently. Each node that is already included in the tree carries a cone of valid growth directions (see Figure 6). When a leaf of the tree is activated it includes any neighbors into the tree which are not part of the tree yet and lie within the cone. A cone is defined as an interval of ports. The cone of the root r contains all six ports. When a new particle q is included in the tree, then its parent p assigns a particular cone of directions to q. Assume parent p has cone c and that q is connected to p via port i of p. By definition $i \in c$, since otherwise p would not include q into the tree. We intersect c with the 120°-cone $[i-1, i+1]$ and pass the resulting cone c' on to q. (Recall that the arithmetic operations on the ports are performed modulo 6.) When doing so we translate c' into the local coordinate system of q such that the cone always includes the same global directions. This simple rule for cone assignments grows an SP-tree in the visibility region of the root r and it does so in a linear number of rounds.

▶ **Lemma 4.** *Given a particle configuration $\mathcal{P}$ with n particles which is $\mathcal{P}$-visible from a particle $r \in \mathcal{P}$, we can grow an SP-tree in $\mathcal{P}$ from r using $O(n)$ rounds.*

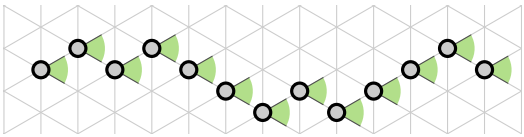


Figure 5 Angle monotone path: for every particle, the remainder of the path lies in a 60°-cone.

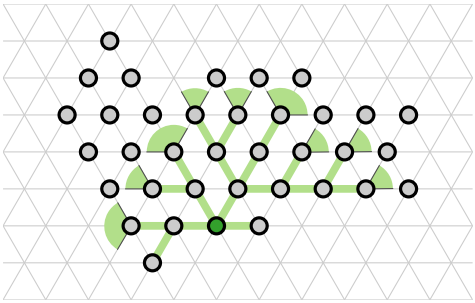


Figure 6 Growing an SP-tree using cones of directions.

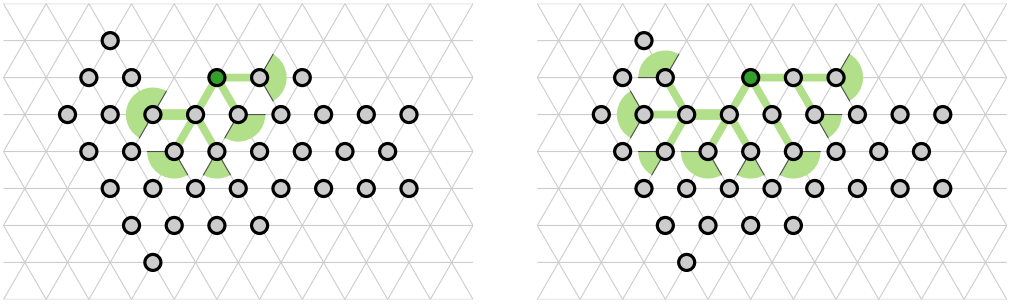


Figure 7 SP-trees with cone extensions. Left: The leftmost particle just extended its cone to 180°. Right: A couple of activations further.

Proof. We are growing the SP-tree from r using the algorithm described above. First of all, observe that any path in G which uses only two adjacent directions from the possible six, is a shortest path. Moreover, every shortest path between two particles p and q uses the same at most two directions. Consider the path π in the SP-tree from r to a particle p. By construction, all particles on π, except for r, have cones of at most 120° (which equals three adjacent directions). Assume that π leaves r in the global direction d_1. As soon as π deviates from d_1 in direction d_2 adjacent to d_1, the cone of available directions shrinks to 60° and contains only the two directions d_1 and d_2. Hence π uses at most two adjacent directions and is therefore a shortest path.

It remains to show that for any particle q in $\mathcal{P}$ there is a path in the SP-tree from r to q. Suppose that q is not part of the SP-tree. Consider a shortest path σ from r to q in $\mathcal{P}$, let p be the last particle on this path that belongs to the SP-tree, and q' be the next particle along σ. We argue that q' lies in one of the directions of the cone of particle p. Indeed, as observed above, path σ consists of at most two adjacent directions. The path π from r to p in the SP-tree uses the same or a subset of the directions of σ, and thus the cone of p contains the direction towards q'. Contradiction. ◀

We now extend our basic algorithm to arbitrary simply-connected particle systems using the shortest-path map SPM(r). The SP-tree constructed by the basic algorithm contains exactly the particles of the visibility region R_0 of SPM(r). As any shortest path from a window particle to r passes through the root of the window, any window incident to R_0 forms a single branch of the SP-tree. To continue the growth of the tree in the other regions of the SPM(r), we extend the cone of valid directions for the root particles of the regions of SPM(r) by 120°. A particle p can detect whether it is a root of an SPM(r)-region by checking its local neighborhood. Specifically, let the parent of p lie in the direction of the port $i+3$. If the cone assigned to p by its parent is $[i-1, i]$ (or $[i, i+1]$), the neighboring node of p in the direction $i+2$ (or $i-2$) is empty, and the node in the direction $i+1$ (or $i-1$) is not empty, then p is the root of an SPM(r)-region, and thus p extends its cone to $[i-1, i+2]$ (or $[i-2, i+1]$) (see Figure 7). Note that if a cone is extended then it becomes a 180°-cone.

▶ **Lemma 5.** *Given a simply-connected particle system $\mathcal{P}$ and a particle $r \in \mathcal{P}$, let* SPM(r) *be the shortest-path map of r in $\mathcal{P}$. A particle $u \in \mathcal{P}$ extends its cone during the construction of an SP-tree if and only if it is the root of a region in* SPM(r).

Proof. Let w be a root of an SPM(r)-region. We first argue that w will extend its cone, when it is a leaf of a growing SP-tree and is activated. Let w lie in an SPM(r)-region with root u (as in Figure 3). Any shortest path from u to w in $\mathcal{P}$ uses exactly two adjacent

directions, one of which is directed along the window of w. Otherwise either w and the particles of the window would not be $\mathcal{P}$-visible to u, or all neighbors of w would be $\mathcal{P}$-visible to u. W.l.o.g., let these two directions be i and $i-1$ in the local coordinate system of w, and let i be the direction along the window. All shortest paths from w to u must go in direction $i+3$, otherwise again all neighbors of w would be $\mathcal{P}$-visible to u. Then, w must have a neighboring particle in the direction of $i+1$, and the neighboring node in the direction $i+2$ must be empty. Otherwise, w would not be part of a window, or it would not be a root of the window. Thus, w will extend its cone.

Let u be a leaf of a growing SP-tree which extends its cone. We now argue that u must be a root of an SPM(r)-region. Let R_0 be the visibility region of r in $\mathcal{P}$. We first assume that $u \in R_0$. Let the parent p of u lie in the direction of port $i+3$. W.l.o.g., assume that u extends its cone $[i-1, i]$ to $[i-1, i+2]$, and thus the neighboring node at port $i+1$ is non-empty, and the neighboring node at port $i+2$ is empty. Let v be the neighbor of u in the direction $i+1$. Consider a maximal chain of particles W' in $\mathcal{P}$ from u in the direction of port i. Particles W' are $\mathcal{P}$-visible from r as a shortest path from r to any $q \in W'$ uses only two directions $i-1$ and i. Particle v is not $\mathcal{P}$-visible from r, as any path from r to v must cross W', and thus use an extra direction $i+1$ or $i+2$. Consider the connected component $\mathcal{R}_v$ of $\mathcal{P} \setminus R_0$ containing v. Since u is adjacent to v it is part of some window W of $\mathcal{R}_v$. The parent p of u is not adjacent to $\mathcal{R}_v$. Since all shortest paths from r to a particle in W pass through u, u must be the root of W. Since u lies on the boundary of $\mathcal{R}_v$, growing the SP-tree further from u is equivalent to growing a new SP-tree only in $\mathcal{R}_v$ with u as the root. Hence the same argument applies recursively. ◀

Lemmas 4 and 5 together imply Theorem 6.

▶ **Theorem 6.** *Given a simply-connected particle configuration $\mathcal{P}$ with n particles and a particle $r \in \mathcal{P}$ we can grow an SP-tree in $\mathcal{P}$ from r using $O(n)$ rounds.*

3 Conclusion

We introduced SP-trees as a new primitive for the Amoebot model and showed how to construct SP-trees in simply-connected particle configurations efficiently. SP-trees can be used to solve the shape reconfiguration problem more efficiently. The *shape* of a particle configuration $\mathcal{P}$ is the set of all nodes in $G_{\mathcal{P}}$. An instance of the shape reconfiguration problem requires a particle configuration $\mathcal{P}$ with shape S to form a target shape T. In ongoing work-in-progress we use multiple SP-trees to organize "supply" particles and "demand" locations, and route supply particles to demand locations efficiently.

References

1 Sarah Cannon, Joshua J. Daymude, Dana Randall, and Andréa W. Richa. A Markov Chain Algorithm for Compression in Self-Organizing Particle Systems. In *Proc. ACM Symposium on Principles of Distributed Computing (PODC)*, pages 279–288, 2016. doi:10.1145/2933057.2933107.

2 Kenneth C. Cheung, Erik D. Demaine, Jonathan R. Bachrach, and Saul Griffith. Programmable Assembly With Universally Foldable Strings (Moteins). *IEEE Transactions on Robotics*, 27(4):718–729, 2011. doi:10.1109/TRO.2011.2132951.

3 Joseph C. Culberson and Robert A. Reckhow. Dent Diagrams: A Unified Approach to Polygon Covering Problems. Technical report, University of Alberta, 1987. TR 87–14.

4 Joshua J. Daymude, Robert Gmyr, Kristian Hinnenthal, Irina Kostitsyna, Christian Scheideler, and Andréa W. Richa. Convex Hull Formation for Programmable Matter. In *Proc. 21st International Conference on Distributed Computing and Networking*, pages 1–10, 2020. doi:10.1145/3369740.3372916.

5 Joshua J. Daymude, Kristian Hinnenthal, Andréa W. Richa, and Christian Scheideler. Computing by Programmable Particles. In Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro, editors, *Distributed Computing by Mobile Entities*, LNCS 11340, chapter 22, pages 615–681. Springer, 2019. doi:10.1007/978-3-030-11072-7_22.

6 Hooman R. Dehkordi, Fabrizio Frati, and Joachim Gudmundsson. Increasing-Chord Graphs On Point Sets. In *Proc. International Symposium on Graph Drawing (GD)*, LNCS 8871, pages 464–475, 2014. doi:10.1007/978-3-662-45803-7_39.

7 Zahra Derakhshandeh, Shlomi Dolev, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. Brief announcement: Amoebot—A New Model for Programmable Matter. In *Proc. 26th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 220–222, 2014. doi:10.1145/2612669.2612712.

8 Zahra Derakhshandeh, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. Universal Shape Formation for Programmable Matter. In *Proc. 28th Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 289–299, 2016. doi:10.1145/2935764.2935784.

9 Zahra Derakhshandeh, Robert Gmyr, Thim Strothmann, Rida Bazzi, Andréa W. Richa, and Christian Scheideler. Leader Election and Shape Formation with Self-organizing Programmable Matter. In *Proc. International Workshop on DNA-Based Computing (DNA)*, LNCS 9211, pages 117–132, 2015. doi:10.1007/978-3-319-21999-8_8.

10 Giuseppe A. Di Luna, Paola Flocchini, Nicola Santoro, Giovanni Viglietta, and Yukiko Yamauchi. Shape formation by programmable particles. *Distributed Computing*, 33:69–101, 2020. doi:10.1007/s00446-019-00350-6.

11 Subir Kumar Ghosh. *Visibility Algorithms in the Plane*. Cambridge University Press, 2007. doi:10.1017/CBO9780511543340.

12 Robert Gmyr, Kristian Hinnenthal, Irina Kostitsyna, Fabian Kuhn, Dorian Rudolph, Christian Scheideler, and Thim Strothmann. Forming Tile Shapes with Simple Robots. In *Proc. International Conference on DNA Computing and Molecular Programming (DNA)*, pages 122–138, 2018. doi:10.1007/978-3-030-00030-1_8.

13 Joseph S. B. Mitchell. A new algorithm for shortest paths among obstacles in the plane. *Annals of Mathematics and Artificial Intelligence*, 3(1):83–105, 1991. doi:10.1007/BF01530888.

14 Matthew J. Patitz. An introduction to tile-based self-assembly and a survey of recent results. *Natural Computing*, 13(2):195–224, 2014. doi:10.1007/s11047-013-9379-4.

15 Alexandra Porter and Andrea Richa. Collaborative Computation in Self-organizing Particle Systems. In *Proc. International Conference on Unconventional Computation and Natural Computation (UCNC)*, LNCS 10867, pages 188–203, 2018. doi:10.1007/978-3-319-92435-9_14.

16 Damien Woods, Ho-Lin Chen, Scott Goodfriend, Nadine Dabby, Erik Winfree, and Peng Yin. Active self-assembly of algorithmic shapes and patterns in polylogarithmic time. In *Proc. 4th Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 353–354, 2013. doi:10.1145/2422436.2422476.

Long plane trees*

Sergio Cabello[†1,2], Michael Hoffmann[‡3], Katharina Klost[4], Wolfgang Mulzer[§4], and Josef Tkadlec[5]

1 **Faculty of Mathematics and Physics, University of Ljubljana, Slovenia**
`sergio.cabello@fmf.uni-lj.si`
2 **Institute of Mathematics, Physics and Mechanics, Slovenia**
3 **Department of Computer Science, ETH Zürich, Switzerland**
`hoffmann@inf.ethz.ch`
4 **Institut für Informatik, Freie Universität Berlin, Germany**
`{kathklost,mulzer}@inf.fu-berlin.de`
5 **Harvard University, USA**
`pepa_tkadlec@fas.harvard.edu`

Abstract

Let $\mathcal{P}$ be a finite point set in the plane in general position. For any spanning tree T on $\mathcal{P}$, we denote by $|T|$ the Euclidean length of T. Let T_{OPT} be a plane (noncrossing) spanning tree of maximum length for $\mathcal{P}$. It is not known whether such a tree can be found in polynomial time. Thus, past research has focused on designing polynomial-time approximation algorithms, typically based on trees of small (unweighted) diameter. We extend this line of research and show how to construct in polynomial time a plane tree T_{ALG} on $\mathcal{P}$ such that T_{ALG} has diameter at most four and $|T_{\text{ALG}}| > 0.546 \cdot |T_{\text{OPT}}|$. This improves substantially over the currently best known approximation factor. Furthermore, we consider the special case of a long plane spanning tree with diameter at most three, and we show that it can be found in polynomial time.

Related Version: A full version is available at `https://arxiv.org/abs/2101.00445`.

1 Introduction

Geometric network design is a common and well-studied task in computational geometry and combinatorial optimization [6–9]. In this family of problems, we are given a set $\mathcal{P}$ of points in general position, and our task is to connect $\mathcal{P}$ into a (geometric) graph that has certain favorable properties. There are many possible objective functions and many different constraints that might be imposed on the resulting graph.

Here, we focus on graphs that achieve a large total edge length while at the same time ensuring that the edges do not cross. In many cases, the objective of maximization and the noncrossing constraint are in conflict. If the goal is to minimize the total edge length, like, e.g., in Euclidean minimum spanning trees or the Euclidean TSP, the noncrossing property is often implied by the triangle inequality. In contrast, when maximizing, say, the total edge length of a spanning tree, the resulting graph will most likely contain many crossings. In this

* This research was started at the 3rd DACH Workshop on Arrangements and Drawings, August 19–23, 2019, in Wergenstein (GR), Switzerland, and continued at the 16th European Research Week on Geometric Graphs, November 18–22, 2019, in Strobl, Austria. We thank all participants of the workshops for valuable discussions and for creating a conducive research atmosphere.
† Supported by the Slovenian Research Agency (P1-0297, J1-9109, J1-8130, J1-8155, J1-1693, J1-2452).
‡ Supported by the Swiss National Science Foundation within the collaborative DACH project *Arrangements and Drawings* as SNSF Project 200021E-171681.
§ Supported in part by ERC StG 757609 and by the German Research Foundation within the collaborative DACH project *Arrangements and Drawings* as DFG Project MU 3501/3-1.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

sense, balancing the noncrossing constraint and the maximization objective is an interesting challenge.

We will consider long plane spanning trees: given a point set $\mathcal{P}$ in general position (i.e., no three points are on a common line), we want to find a longest spanning tree on $\mathcal{P}$ such that no two edges cross. The precise complexity for this problem is unknown, but it is conjectured to be NP-hard. This stands in contrast to the greedy polynomial time algorithms for short (necessarily plane) spanning trees and long (possibly not plane) spanning trees.

As a polynomial-time algorithm for the long plane case has eluded researchers for many years, the focus has shifted to approximation algorithms. The first such algorithm, giving a 0.5-approximation, is due to Alon et al. [1]. This approximation factor was then improved to 0.502 by Dumitrescu and Tóth [5]. This result led to a series of improvements from 0.503 [3] over 0.512 [4] to the most recent result of a 0.519-approximation [2].

We substantially increase the approximation factor to a fixed $f > 0.5467$, and we give a polynomial time algorithm for finding a longest tree of unweighted diameter at most 3.

2 Approximation Algorithm

We describe an algorithm to find a plane spanning tree on a given point set $\mathcal{P}$, and we show that the resulting tree is a good approximation for a longest plane spanning tree on $\mathcal{P}$. The algorithm considers two families of trees. The first family consists of *stars*: for a point $a \in \mathcal{P}$, the star S_a rooted at a is the tree that connects all points $p \in \mathcal{P} \setminus \{a\}$ to a.

The second family of trees $T_{a,b}$ is parameterized by two distinct points $a, b \in \mathcal{P}$. The trees $T_{a,b}$ are defined as follows: let $\mathcal{P}_a$ be the points of $\mathcal{P}$ that are closer to a than to b, and let $\mathcal{P}_b = \mathcal{P} \setminus \mathcal{P}_a$. We connect a to every point in $\mathcal{P}_b$, and then we connect each point of $\mathcal{P}_a \setminus \{a\}$ to some point of $\mathcal{P}_b$ without introducing crossings. We will not go into the details of the second step here, but it is possible to make these connections in a simple deterministic way. See Figure 1 for an example.

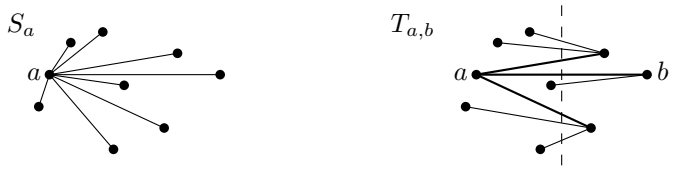


Figure 1 A star S_a and a tree $T_{a,b}$.

The algorithm computes all stars S_a, for $a \in \mathcal{P}$, and all trees $T_{a,b}$, for ordered pairs $(a, b) \in \mathcal{P}^2$ with $a \neq b$. The algorithm returns a longest among all those trees. This process can be implemented in polynomial time, and we call the resulting tree T_{ALG}.

▶ **Theorem 2.1.** *For any finite planar point set $\mathcal{P}$ in general position, the tree T_{ALG} has Euclidean length at least $f \cdot |T_{\text{OPT}}|$, where $|T_{\text{OPT}}|$ denotes the length of a longest plane tree on $\mathcal{P}$ and $f > 0.5467$ is the fourth smallest real root of the polynomial*

$$P(x) = -80 + 128x + 504x^2 - 768x^3 - 845x^4 + 1096x^5 + 256x^6.$$

Proof sketch. *Particularly towards the end of this proof sketch, we omit some details due to space constraints. The detailed calculations can be found in the full version.*

We fix some assumptions on $\mathcal{P}$. First, we assume that $\mathcal{P}$ has diameter exactly 2. Next, fix some optimal tree T_{OPT} on $\mathcal{P}$ and a longest edge ab of T_{OPT}. Write the Euclidean length

of ab as $\|ab\| = 2d$. Let $D(a, 2)$ and $D(b, 2)$ be the disks with radius 2 and centers a and b, respectively. As $\mathcal{P}$ has diameter 2, all points lie inside the lens $D(a, 2) \cap D(b, 2)$. Without loss of generality, we assume that $a = (-d, 0)$ and $b = (d, 0)$.

From arguments established in earlier research [4, Lemma 2.1], we can conclude that if $2d \leq 1/f$, the result follows. Hence, from now on, we focus on the case $2d > 1/f$.

Let s, s' be the points on the y-axis with $\|sa\| = \|sb\| = \|s'a\| = \|s'b\| = 2df$, where s is the one with the positive y-coordinate. Since $2df > 1$, the circles $\partial D(s, 2df)$ and $\partial D(s', 2df)$ intersect the boundary of the lens. We call the region that is inside the lens but above or below both circles the *far region* (shaded green in Figure 2), and the remaining part of the lens is referred to as the *truncated lens* (shaded yellow in Figure 2).

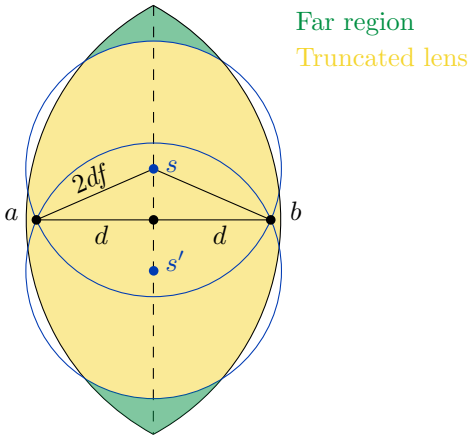


Figure 2 The lens is split into the far region (green) and the truncated lens (yellow).

Suppose that there is a point $c \in \mathcal{P}$ in the far region. Then, we argue that a longest of the stars S_a, S_b, and S_c is a good approximation. Denote by R the circumradius of the triangle $\triangle abc$, and let g be the center of mass of $\mathcal{P} \setminus \{a, b, c\}$. Then, as $\triangle abc$ is acute-angled, there is one point q among a, b, c such that $\|gq\| \geq R$. Having fixed q, the triangle inequality gives $\sum_{p \in \mathcal{P} \setminus \{a,b,c\}} \|pq\| \geq (n-3) \cdot R$. Together with $\|qa\| + \|qb\| + \|qc\| \geq 2R$, this yields $|S_q| = \sum_{p \in \mathcal{P}} \|pq\| \geq (n-1) \cdot R \geq f \cdot (n-1) \cdot 2d \geq f \cdot |T_{\text{OPT}}|$.

Hence, from now on, we can assume that $2d > 1/f$ and that the far region contains no point from $\mathcal{P}$. Consider the five trees $T_{a,b}$, $T_{b,a}$, S_a, S_b, and T_{OPT}. They all contain the edge ab. We conceptually direct the other edges towards ab and, given a point $p \in \mathcal{P}$ and a tree T, denote the length of the outgoing edge from p in T by $\ell_T(p)$. Given a real parameter $\beta \in (0, 1/2)$ we define the weighted average of the lengths of the edges assigned to a point p over the first four trees as:

$$\text{avg}(p, \beta) = (1/2 - \beta) \cdot \ell_{S_a}(p) + \beta \cdot \ell_{T_{a,b}}(p) + \beta \cdot \ell_{T_{b,a}}(p) + (1/2 - \beta) \cdot \ell_{S_b}(p)$$

Summing up over all $p \in \mathcal{P} \setminus \{a, b\}$ and adding the length of the edge ab yields the weighted average of the length of all four trees. Note that a longest of the four trees is guaranteed to be longer than the weighted average. This reduces our problem to that of finding β such that for each point in $\mathcal{P} \setminus \{a, b\}$ we have:

$$\text{avg}(p, \beta) \geq f \cdot \ell_{T_{\text{OPT}}}(p). \tag{1}$$

In the following we assume without loss of generality that $p = (x, y)$ with $x, y \geq 0$. Let p_a be the point with x-coordinate $-(2 - d)$ on the ray pa. Furthermore, if the x-coordinate

of p is less than d, let p_b be the point with x-coordinate $2-d$ on the ray pb. When the x-coordinate of p is at least d, then the ray pb does not intersect the vertical line with x-coordinate $2-d$ and we set $p_b = b$. Additionally, define p_u to be the furthest point from p on the portion of the boundary of the far region that is contained in the circle $k = \partial D(s, 2df)$. See Figure 3 for a visualization. We claim that

$$\ell_{T_{\mathrm{OPT}}}(p) \leq \min\{2d, \max\{\|pp_a\|, \|pp_u\|\}\}. \tag{2}$$

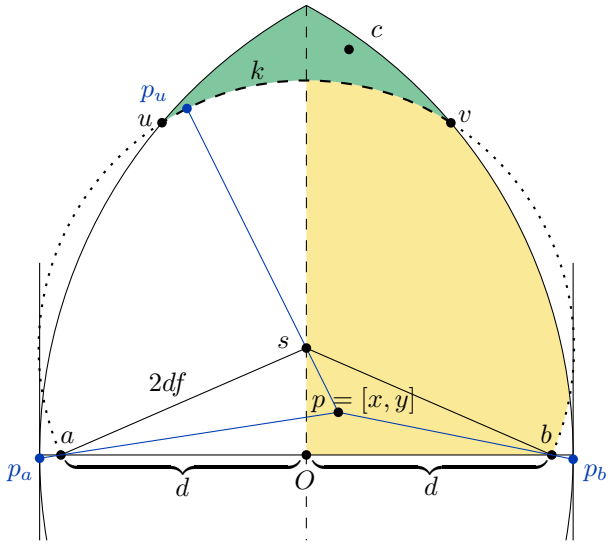


Figure 3 The definition of the special points

To show (2), first note that if $\|pp_a\| \geq 2d$ then we are done as the right-hand side becomes $2d$ and the left-hand side is at most $2d$ by assumption. Next, tedious algebra shows that if $\|pp_a\| \leq 2d$ then $\|pp_b\| \leq \|pp_a\|$. The rest follows by denoting by p_f the point in the truncated lens furthest from p and doing a case distinction over the quadrant containing p_f:

1. If p_f lies in the third or fourth quadrant then $\|pp_a\|$ is larger than $\|pp_f\|$.
2. If p_f lies in the first quadrant, mirroring p_f along the y-axis gives a point further away from p than p_f.
3. If p_f lies in the second quadrant then it lies on the boundary of the truncated lens. Let t be the topmost point in the truncated lens and let u be the to higher intersection point of $\partial D(s, 2df)$ and $\partial D(b, 2)$, see Figure 4 for an illustration. If p_f lies on the arc tu and $p \in N$ we are done, as $p_u = u$. If $p \in S$ we are done by triangle inequality. Finally if p_f lies on the arc ul we have $\|pp_f\| \leq \max\{\|pl\|, \|pu\|\}$ and are again done, as $\|pl\| \leq \|pp_a\|$ and $\|pu\| \leq \|pp_u\|$.

Combining (1) and (2), it suffices to find $\beta \in [0, 1/2]$ such that both $\mathrm{avg}(p, \beta) \geq f \cdot \min\{2d, \|pp_a\|\}$ and $\mathrm{avg}(p, \beta) \geq f \cdot \min\{2d, \|pp_u\|\}$. To find such β, our main technical insight is the following lower bound on $\mathrm{avg}(p, \beta)$ that holds for any $\beta \in [0, 1/2]$:

$$\mathrm{avg}(p, \beta) \geq \frac{d \cdot (1-\beta) + x \cdot 2\beta}{d + x}. \tag{3}$$

To prove (3), we start with unpacking the definition:

$$\mathrm{avg}(p, \beta) \geq (1/2 - \beta) \cdot \|pa\| + \beta \cdot \|pa\| + \beta \cdot x + (1/2 - \beta) \cdot \|pb\|.$$

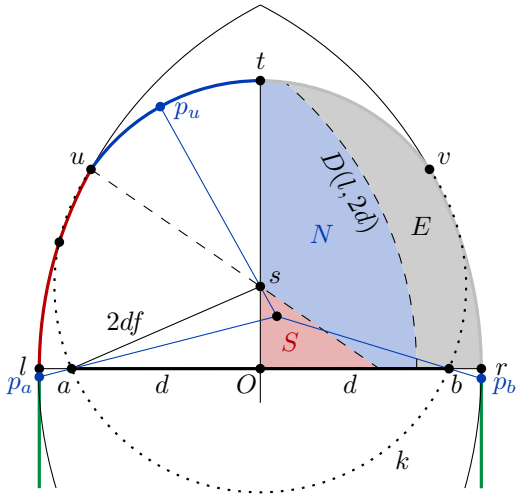


Figure 4 For $p \in E$ we have $\ell_{T_{\mathrm{OPT}}}(p) \leq 2d$, for $p \in N$ we have $\ell_{T_{\mathrm{OPT}}}(p) \leq \max\{\|pp_a\|, \|pu\|\}$ and for $p \in S$ we have $\ell_{T_{\mathrm{OPT}}}(p) \leq \max\{\|pp_a\|, \|pp_u\|\}$.

Let $p' = (-x, y)$ be the reflection of p about the y axis (see Figure 5). Triangle inequality $\|p'p\| + \|pb\| \geq \|p'b\| = \|pa\|$ yields $\beta \cdot x = \frac{1}{2}\beta \cdot \|p'p\| \geq \frac{1}{2}\beta \cdot (\|pa\| - \|pb\|)$ and we get

$$\mathrm{avg}(p, \beta) \geq \frac{1}{2} \cdot \|pa\| + \frac{1}{2}\beta \cdot \|pa\| + \left(\frac{1}{2} - \frac{3}{2}\beta\right) \cdot \|pb\|.$$

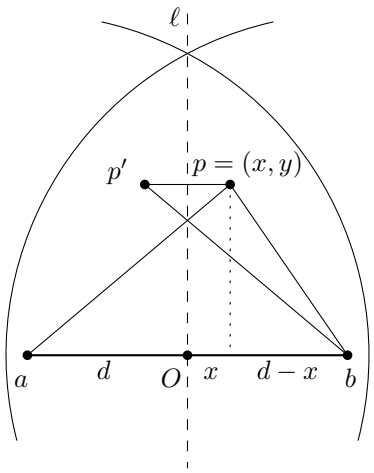


Figure 5 Mirroring p along the y-axis.

Next we claim that $\|pb\| \geq \frac{d-x}{d+x} \cdot \|pa\|$: Indeed, upon squaring, using the Pythagorean theorem and clearing the denominators this becomes $y^2 \cdot 4dx \geq 0$ which is true. Using this bound on the term containing $\|pb\|$, the inequality (3) is proved as follows:

$$\mathrm{avg}(p, \beta) \geq \frac{(1+\beta)(d+x) + (1-3\beta)(d-x)}{2(d+x)} \cdot \|pa\| = \frac{(1-\beta) \cdot d + 2\beta \cdot x}{d+x} \cdot \|pa\|.$$

Using (3) and some elementary geometric considerations (briefly sketched below) we can prove the following claims for any point $p = (x, y)$ with $x, y \geq 0$:

1. We have $\operatorname{avg}(p,\beta) \geq f \cdot \min\{2d, \|pp_a\|\}$, provided that

$$\frac{2f-1}{5-8f} \leq \beta \leq \frac{1}{2} \cdot f. \tag{4}$$

2. We have $\operatorname{avg}(p,\beta) \geq f \cdot \min\{2d, \|pp_u\|\}$, provided that $\beta < \frac{151}{304} \cdot f$ and $\frac{1}{2} \leq f \leq \frac{19}{32}$ and

$$\frac{2f-1}{2\sqrt{5-8f}-1} \leq \beta \leq 1 - f\sqrt{4f^2-1} - 2f^2. \tag{5}$$

To prove 1., we distinguish the cases $x \geq 3d-2$ and $x < 3d-2$ and show $\operatorname{avg}(p,\beta) \geq f \cdot 2d$ and $\operatorname{avg}(p,\beta) \geq f\|pp_a\|$, respectively. To prove 2., we distinguish the cases $y \leq y(u)$ and $y > y(u)$. In the first case we use that the left-hand side of (3) is increasing in y, while $\|pp_u\|$ is decreasing. The second case essentially follows from basic algebra and the Pythagorean theorem.

Finally, it is straightforward (although again tedious) to check that for our definition of f the left-hand side and the right-hand side of the constraint (5) are equal. To be precise, after setting the left-hand side and the right-hand site equal and applying some algebra, which includes two times the squaring of both sites, yields the polynomial given in the statement of the theorem. Our choice of f is not only a solution to the origin polynomial but also yields a value value $\beta^\star \doteq 0.1604$ such that setting $\beta = \beta^\star$ satisfies also (4) and the other constraints in 2. Note that for any $f' > f$, the two constraints on β from (5) are contradictory, except for $f' = \frac{5}{8}$ when they reduce to $-\frac{1}{4} \leq \beta \leq -\frac{1}{4}$ which is not a permissible value of β. ◀

3 Polynomial-time algorithm for diameter 3

In this section we describe a polynomial time algorithm for finding a longest plane spanning tree of diameter at most 3 for a given set of points in general position. Note that all trees of diameter at most 3 have a cut edge ab whose removal decomposes the tree into at most 2 stars, one rooted at a and one rooted at b. We also call such trees *bistars* rooted at a and b.

▶ **Theorem 3.1.** *We can find a longest bistar on a point set $\mathcal{P}$ in polynomial time.*

Proof. We argue how to find the best bistar with fixed roots a, b in polynomial time. The statement of the theorem then follows by iterating over all possible pairs of roots.

Without loss of generality we assume that ab is a horizontal edge and that all points lie above ab. We use dynamic programming. We call a pair $p, q \in \mathcal{P}$ a *valid pair*, if ap and bq do not cross. Let $Q(p,q)$ be the quadrilateral defined by ab, ap, bq and the horizontal line through $\min\{y(p), y(q)\}$ as shown in Figure 6.

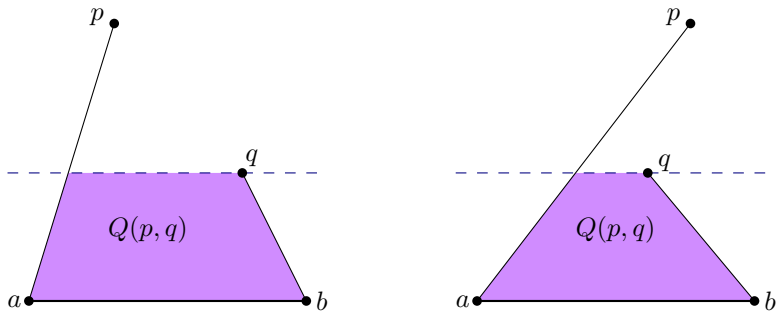


Figure 6 Two examples of valid pairs p, q with their quadrilaterals $Q(p,q)$ shaded.

We denote by $Z(p,q)$ the length of the best bistar rooted at a and b on the points in the interior of $Q(p,q)$. Let $k_{p,q}$ be the highest point in $Q(p,q)$ and let $L_{p,q}$ and $R_{p,q}$ be the regions in $Q(p,q)$ to the left of $ak_{p,q}$ and to the right of $bk_{p,q}$, respectively (see Figure 7).

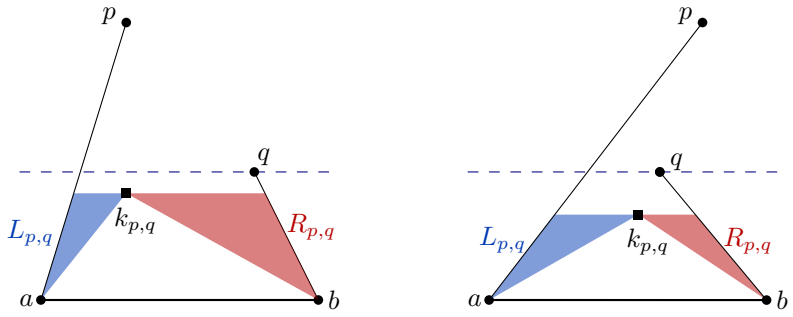


Figure 7 Fixing $k_{p,k}$ gives two possible triangular regions where edges are fixed.

As we aim to compute a bistar, $k_{p,q}$ is either connected to a or to b. In the first case, all points in $L_{p,q}$ have to be connected to a in order to prevent crossings. The points in $Q(k_{p,q},q)$ can now be connected to either a or b, so the best bistar can be found recursively in this region. A symmetric argument holds if $k_{p,q}$ is connected to b. The resulting tree is a bistar rooted at a and b by induction. Formally, for each valid pair p,q we have the following recurrence:

$$Z(p,q) = \begin{cases} 0 & \text{if no point of } \mathcal{P} \text{ is in } Q(p,q), \\ \max \begin{cases} Z(k_{p,q},q) + \|ak_{p,q}\| + \sum_{l\in L_{p,q}} \|al\| \\ Z(p,k_{p,q}) + \|bk_{p,q}\| + \sum_{r\in R_{p,q}} \|br\| \end{cases} & \text{otherwise.} \end{cases}$$

Let p,q be a valid pair, let L_p be all points to the left of the line through ap and let R_q be all points to the right of the line through bq. Then all values of the form

$$\left(\sum_{l\in L_p} \|al\|\right) + \left(\sum_{r\in R_q} \|br\|\right) + \|ap\| + \|bq\| + \|ab\| + Z(p,q), \tag{6}$$

describe the length of a plane, not necessarily spanning, bistar rooted at a and b.

Now assume that a is connected to the point with maximum y-coordinate. Furthermore let q^* be the highest point connected to b and p^* be the point with smallest angle at a above q^*. Then p^*, q^* is a valid pair and the length of the optimal bistar is obtained by plugging p^* and q^* into (6). Now by taking the maximum of (6) for all valid pairs, together with the maximum of S_a and S_b, we obtain a longest bistar rooted at a and b.

The recurrence can be implemented in polynomial time using a standard dynamic programming approach. ◀

In the full version of the paper, we give details on the implementation and show that the dynamic program can be implemented in $\mathcal{O}(n^2)$ time.

4 Conclusion

We showed an $f \geq 0.5467$-approximation algorithm for the longest plane spanning tree problem. Furthermore we gave a polynomial time algorithm to solve the problem for fixed diameter at most 3.

There are some related open questions. First, is the factor f we obtained tight for our algorithm? While all the steps in the analysis are tight, the overall analysis might not be. Second, how well does the optimal tree of diameter 3 approximate the optimal tree of arbitrary diameter? This is especially interesting, as we know of point sets with longest trees of diameter $\Theta(n)$. Finally, there are the broader questions of finding a PTAS and settling the complexity.

References

1 Noga Alon, Sridhar Rajagopalan, and Subhash Suri. Long non-crossing configurations in the plane. *Fundam. Inform.*, 22(4):385–394, 1995. doi:10.3233/FI-1995-2245.

2 Ahmad Biniaz. Euclidean bottleneck bounded-degree spanning tree ratios. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 826–836. SIAM, 2020. doi:10.1137/1.9781611975994.50.

3 Ahmad Biniaz, Prosenjit Bose, Kimberly Crosbie, Jean-Lou De Carufel, David Eppstein, Anil Maheshwari, and Michiel Smid. Maximum plane trees in multipartite geometric graphs. *Algorithmica*, 81(4):1512–1534, 2019. doi:10.1007/s00453-018-0482-x.

4 Sergio Cabello, Aruni Choudhary, Michael Hoffmann, Katharina Klost, Meghana M Reddy, Wolfgang Mulzer, Felix Schröder, and Josef Tkadlec. A better approximation for longest noncrossing spanning trees. In *36th European Workshop on Computational Geometry (EuroCG)*, 2020.

5 Adrian Dumitrescu and Csaba D. Tóth. Long non-crossing configurations in the plane. *Discrete Comput. Geom.*, 44(4):727–752, 2010. doi:10.1007/s00454-010-9277-9.

6 David Eppstein. Spanning trees and spanners. In Jörg-Rüdiger Sack and Jorge Urrutia, editors, *Handbook of Computational Geometry*, pages 425–461. North Holland / Elsevier, 2000. doi:10.1016/b978-044482537-7/50010-3.

7 Sariel Har-Peled. *Geometric approximation algorithms*, volume 173 of *Mathematical Surveys and Monographs*. American Mathematical Society, Providence, RI, 2011. doi:10.1090/surv/173.

8 Joseph S. B. Mitchell. Shortest paths and networks. In Jacob E. Goodman, Joseph O'Rourke, and Csaba D. Tóth, editors, *Handbook of Discrete and Computational Geometry*, chapter 31, pages 811–848. CRC Press, Boca Raton, 3rd edition, 2017. doi:10.1201/9781315119601.

9 Joseph S. B. Mitchell and Wolfgang Mulzer. Proximity algorithms. In Jacob E. Goodman, Joseph O'Rourke, and Csaba D. Tóth, editors, *Handbook of Discrete and Computational Geometry*, chapter 32, pages 849–874. CRC Press, Boca Raton, 3rd edition, 2017. doi:10.1201/9781315119601.

Terrain prickliness: theoretical grounds for low complexity viewsheds

Ankush Acharyya[1], Ramesh K. Jallu[1], Maarten Löffler[2], Gert G.T. Meijer[2], Maria Saumell[1,3], Rodrigo I. Silveira[4], Frank Staals[2], and Hans Raj Tiwary[5]

1 **The Czech Academy of Sciences, Institute of Computer Science, Czech Republic**
{acharyya,jallu,saumell}@cs.cas.cz
2 **Department of Information and Computing Sciences, Utrecht University, Netherlands**
m.loffler@uu.nl, g.t.meijer@students.uu.nl
3 **Department of Theoretical Computer Science, Faculty of Information Technology, Czech Technical University in Prague, Czech Republic**
4 **Department de Matemàtiques, Universitat Politècnica de Catalunya, Spain**
rodrigo.silveira@upc.edu
5 **Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University, Czech Republic**
hansraj@kam.mff.cuni.cz

1 Introduction

Two points on a triangulated terrain are mutually *visible* if the straight line segment connecting them does not intersect the interior of the terrain; the set of all points visible from a given point p is the *viewshed* of p. Viewsheds are useful, for example, in evaluating the visual impact of potential constructions [2], analyzing the coverage of an area by fire watchtowers [8], or measuring the scenic beauty of a landscape [14]. In a 1.5D terrain, the viewshed of one viewpoint can have a complexity that is linear on the number of vertices, and can be computed in linear time [7]. In contrast, in 2.5D terrains the complexity of the viewshed of one viewpoint can be quadratic in the number of vertices [11], which makes its computation too slow for most practical uses when terrains are large. A typical quadratic construction is shown schematically in Fig. 1. Viewsheds with quadratic complexity are uncommon in real terrains, and there have been attempts to define theoretical conditions for a terrain to be "realistic" that guarantee, among others, that viewsheds cannot be that large [3, 13]. In this paper we introduce a new terrain measure, based on the idea that viewsheds are more complex in "rugged" terrains [9], and study its relation to viewshed complexity.

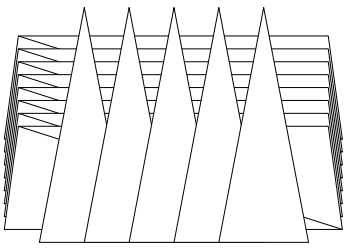


Figure 1 If a viewpoint is placed in front of the visible triangles and very far from them, its viewshed has quadratic complexity.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

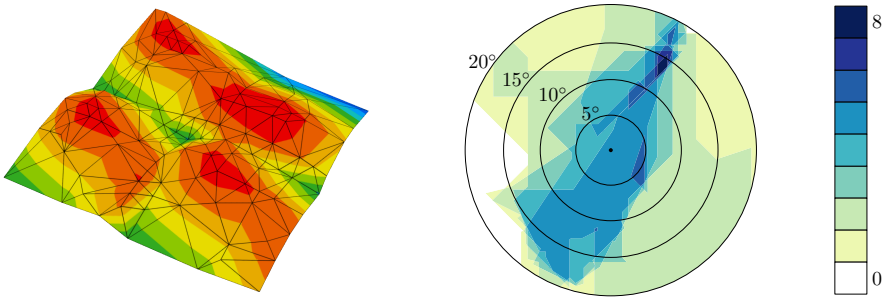


Figure 2 (left) A polyhedral terrain T. Height (z-coordinates) are indicated using colors. (right) A visualization of the prickliness $\pi_{\vec{v}}(T)$ as a function of $\vec{v}$, in degrees from vertical $(0, 0, 1)$. The maximum prickliness is 8, attained at a direction about 13° north-east from the origin.

Let T be a polyhedral surface and let $\vec{v}$ be a vector. We define $\pi_{\vec{v}}(T)$ to be the number of internal vertices of T that are extremal in direction $\vec{v}$. Then we define the *prickliness* of T, $\pi(T)$, to be the maximum number of local maxima over all directions; that is,

$$\pi(T) = \max_{\vec{v}} \pi_{\vec{v}}(T).$$

We say that T is a *terrain* if any vertical line intersects T at most once. If T is a terrain, for any $\vec{v}$ with a negative z-coordinate we set $\pi_{\vec{v}}(T) = 0$ by definition. Fig. 2 shows a small terrain and the resulting prickliness, shown as a function of the direction of $\vec{v}$.

In Section 2, we investigate the correlation between the prickliness of a terrain and the maximum viewshed complexity. In Section 3 we investigate the computational problem of calculating the prickliness of a terrain. Finally, in Section 4, we report on experiments that measure the prickliness of real terrains, and confirm the correlation between prickliness and viewshed complexity in practice. Omitted proofs can be found in the full version of this paper [1].

2 Prickliness and viewshed complexity

We first consider 1.5D terrains. In this case, prickliness is not related to viewshed complexity.

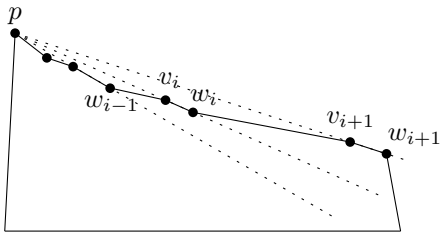


Figure 3 A 1.5D terrain with constant prickliness but linear viewshed complexity.

▶ **Theorem 2.1.** *There exists a 1.5D terrain T with n vertices and constant prickliness, and a viewpoint on T with viewshed complexity $\Theta(n)$. (Refer to Fig. 3.)*

Surprisingly, and in contrast to Theorem 2.1, we will show in Theorem 2.4 that in 2.5D there is a provable relation between prickliness and viewshed complexity. We recall some terminology introduced in [6]. Let v be a vertex of T, and let p be a viewpoint. We denote

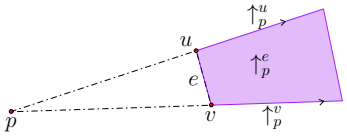


Figure 4 A vase.

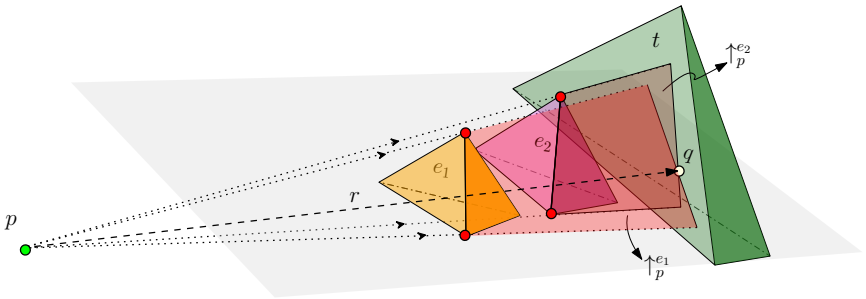


Figure 5 The situation in the proof of Lemma 2.3; q is a vertex of type 3.

by $\uparrow_p^v$ the half-line with origin at p in the direction of vector $\overrightarrow{pv}$. Now, let $e = uv$ be an edge of T. The *vase* of p and e, denoted $\uparrow_p^e$, is the region bounded by e, $\uparrow_p^u$, and $\uparrow_p^v$ (see Fig. 4). Vertices of the viewshed of p can have three types [6]. A vertex of type 1 is a vertex of T, of which there are clearly only n. A vertex of type 2 is the intersection of an edge of T and a vase. A vertex of type 3 is the intersection of a triangle of T and two vases (see Fig. 5 for an example). With the following two lemmas we will be able to prove Theorem 2.4.

▶ **Lemma 2.2.** *There are at most $O(n \cdot \pi(T))$ vertices of type 2.*

Proof. Consider an edge e of T and let H be the plane spanned by e and p. Consider the viewshed of p on e. Let qr be a maximal invisible portion of e surrounded by two visible ones. Since q and r are vertices of type 2, the open segments pq and pr pass through a point of T. On the other hand, for any point x in the open segment qr, there exist points of T above the segment px. This implies that there is a continuous portion of T above H such that the vertical projection onto H of this portion lies on the triangle pqr. Such portion has a local maximum in the direction perpendicular to H which is a convex and internal vertex of T. In consequence, each invisible portion of e surrounded by two visible ones can be assigned to a distinct point of T that is a local maximum in the direction perpendicular to H. Hence, in the viewshed of p, e is partitioned into at most $2\pi(T) + 3$ parts.[1] ◀

▶ **Lemma 2.3.** *There are at most $O(n \cdot \pi(T))$ vertices of type 3.*

Proof. Let q be a vertex of type 3 in the viewshed of p. Point q is the intersection between a triangle t of T and two vases, say, $\uparrow_p^{e_1}$ and $\uparrow_p^{e_2}$; see Fig. 5. Let r be the ray with origin at p and passing through q. Ray r intersects edges e_1 and e_2. First, we suppose that e_1 and e_2 do not share any vertex and, without loss of generality, we assume that $r \cap e_1$ is closer to p than $r \cap e_2$. Notice that $r \cap e_2$ is a vertex of type 2 because it is the intersection of e_2

[1] We obtain $2\pi(T) + 3$ parts when the first and last portion of e are invisible; otherwise, we obtain fewer parts.

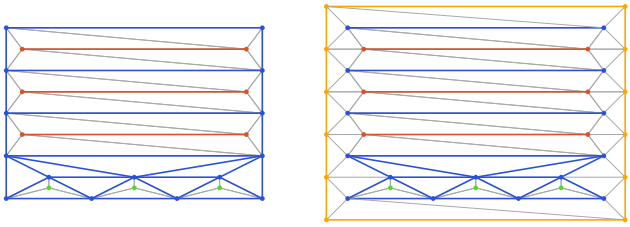


Figure 6 Schematic top-down view of the classic quadratic construction (left), and the same adapted to have small prickliness (right). The camera with quadratic behaviour is at $(0, -\infty)$ (see Fig. 1 for the resulting view). Blue vertices/edges are low, red are medium hight, and green are high. The right construction introduces a new height (yellow) between medium and high, and changes the triangulation slightly, to ensure that all convex vertices in the construction are green.

and $\uparrow_p^{e_1}$, and $\uparrow_p^{e_1}$ partitions e_2 into a visible and an invisible portion. Thus, we charge q to $r \cap e_2$. If another vertex of type 3 was charged to $r \cap e_2$, then such a vertex would also lie on r. However, no point on r after q is visible from p because the visibility is blocked by t. Hence, no other vertex of type 3 is charged to $r \cap e_2$.

If e_1 and e_2 are both incident to a vertex v, since $t \cap \uparrow_p^{e_1} \cap \uparrow_p^{e_2}$ is a type 3 vertex, we have that r passes through v. Therefore, q is the first intersection point between r (which can be seen as the ray with origin at p and passing through v) and the interior of some triangle in T. Therefore, any vertex v of T creates at most a unique vertex of type 3 in this way. ◀

▶ **Theorem 2.4.** *The complexity of a viewshed in a 2.5D terrain is* $O(n \cdot \pi(T))$.

Next we describe a construction showing that the theorem is best possible.

▶ **Theorem 2.5.** *There exists a 2.5D terrain* T *with* n *vertices and prickliness* $\pi(T)$*, and a viewpoint on* T *with viewshed complexity* $\Theta(n \cdot \pi(T))$.

Proof. Consider the standard quadratic viewshed construction, composed of a set of front mountains and back triangles (Fig. 6, left). Notice that there can be at most $\pi(T)$ mountains "at the front". We add a surrounding box around the construction, see Fig. 6 (right), such that each vertex of the back triangles is connected to at least one vertex on this box. We set the elevation of the box so that it is higher than all the vertices of the back triangles, but lower than those of the front mountains. In this way, no vertex of the back triangles is a local maximum in any direction, and all local maxima come from the front. ◀

3 Prickliness computation

In 1.5D, for every internal and convex vertex v in T we consider the set of *feasible unit vectors* $se(v)$, such that $\vec{w} \in se(v)$ if v is a local maximum of T in direction $\vec{w}$. Note that these feasible vectors $\vec{w}$ can be represented as unit vectors, and then the feasible set becomes a region of the unit circle $\mathbb{S}^1$. To find $se(v)$, for each edge e of T incident to v we consider the line ℓ through v which is perpendicular to e. Then we take the half-plane bounded by ℓ and opposite to e, and we translate it so that its boundary contains the origin. Finally, we intersect this half-plane with $\mathbb{S}^1$, which yields a half-circle. We intersect the two half-circles associated to the two edges of T incident to v, and obtain a sector of $\mathbb{S}^1$. For each direction $\vec{w}$ contained in the sector, the two corresponding edges do not extend further than v in direction $\vec{w}$. Thus, v is a local maxima in direction $\vec{w}$ and this sector indeed represents $se(v)$. See Fig. 7 for an example.

Clearly, $se(v)$ can be computed in constant time. The prickliness of T is the maximum number of sectors with non-empty intersection. This can be computed in $O(n \log n)$ time by sorting the bounding angles of the sectors.

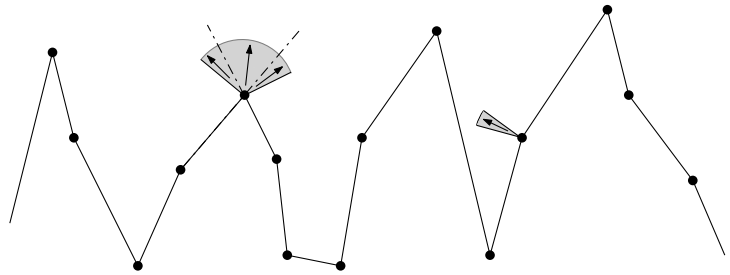


Figure 7 In shaded, $se(v)$ for the corresponding vertices.

▶ **Theorem 3.1.** *In 1.5D, the prickliness can be computed in $O(n \log n)$ time.*

In 2.5D we proceed similarly. For each convex terrain vertex v, consider the cone $co(v)$ of the unit sphere $\mathbb{S}^2$ containing all vectors $\vec{w}$ such that v is a local maximum of T in direction $\vec{w}$. To compute $co(v)$, we consider all edges of T incident to v. Let $e = vu$ be such an edge, and consider the plane orthogonal to e through v. Let H be the half-space which is bounded by this plane and does not contain u. We translate H so that the plane bounding it contains the origin; let H_e be the intersection of the obtained half-space with the unit sphere $\mathbb{S}^2$. Then, for any unit vector $\vec{w}$ in H_e, the edge e does not extend further than v in direction $\vec{w}$. We repeat this for all edges incident to v, and consider the intersection $co(v)$ of all the obtained half-spheres H_e. For any unit vector $\vec{w}$ in $co(v)$, no edge incident to v extends further than v in direction $\vec{w}$. Since v is convex, v is a local maximum in direction $\vec{w}$.

To compute the prickliness we find a unit vector that lies in the maximum number of cones of type $co(v)$. To simplify, rather than considering the cones on the sphere, we extend them until they intersect the boundary of a unit cube $\mathbb{Q}$ centered at the origin. The conic regions of type $co(v)$ intersect the faces of $\mathbb{Q}$ in (overlapping) convex regions. Note that finding a unit vector that lies in the maximum number of regions of type $co(v)$ on $\mathbb{S}$ is equivalent to finding a point on the surface of $\mathbb{Q}$ that lies in the maximum number of extended regions of type $co(v)$. The second problem can be solved by computing the maximum overlap of convex regions using a topological sweep [4], for each face of the cube. Computing the intersection between the extended regions of type $co(v)$ (for all convex vertices v) and the boundary of $\mathbb{Q}$ takes $O(n \log n)$ time, and topological sweep to find the maximum overlap takes $O(n^2)$ time.

▶ **Theorem 3.2.** *In 2.5D, the prickliness can be computed in $O(n^2)$ time.*

3.1 Lower bounds

We begin by showing that $\Omega(n \log n)$ is a lower bound for computing prickliness in 1.5D. The reduction is from element distinctness, which has an $\Omega(n \log n)$ lower bound in the bounded-degree algebraic decision tree model [10]. Let $\mathcal{S} = \{x_1, x_2, \ldots, x_n\}$ be n positive integer elements. We multiply all elements of $\mathcal{S}$ by $180/(\max \mathcal{S} + 1)$ and obtain a new set $\mathcal{S}'$. For each $x'_i \in \mathcal{S}'$, we create a convex vertex v_i with $se(v_i) = [x'_i - \varepsilon, x'_i + \varepsilon]$; refer to Fig. 8.[2]

[2] We sometimes write $se(v) = [\alpha, \beta]$, where α and β are the angles bounding the sector.

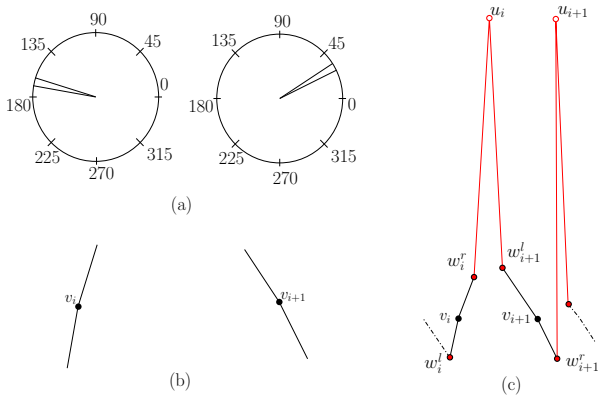


Figure 8 (a) Sectors associated to the element set $\{160, 25\}$. (b) The corresponding convex vertices. (c) Construction of the terrain.

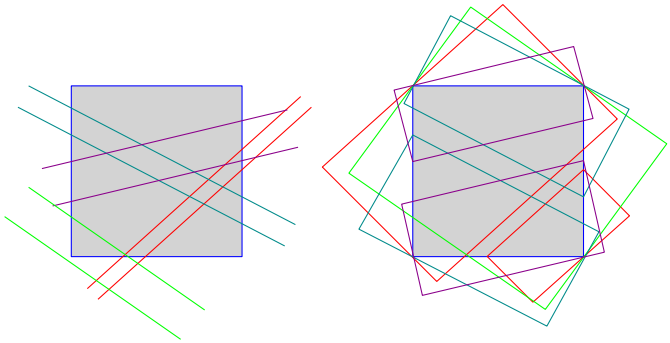


Figure 9 Instance of the problem of coverage with strips (left), and the equivalent problem of coverage with rectangles (right).

We place all v_i at the same height, place two neighbors next to each v_i to satisfy its set of feasible vectors, and separate the resulting mini-chains with additional vertices that are high enough to make the neighbors of v_i concave, and such that their own feasible set covers all sets $se(v_i)$. Then the prickliness of T is n if and only if all elements in $\mathcal{S}$ are distinct.

▶ **Theorem 3.3.** *The problem of computing the prickliness of a 1.5D terrain has an $\Omega(n \log n)$ lower bound in the bounded-degree algebraic decision tree model.*

In 2.5D we show that computing prickliness is 3SUM-hard. The reduction is from covering a square by strips [5]: Given a square Q and m strips of infinite length, does there exist a point in Q not covered by any of these m strips? The complement of each strip is given by two half-planes. Consider the $2m$ half-planes obtained in this way; the question above is equivalent to whether there is a point in Q covered by m of the half-planes. We now replace each half-plane by the smallest rectangle that contains the intersection of Q with the half-plane; see Fig. 9. Let $\mathcal{A}$ be the arrangement containing Q and the rectangles.

▶ Observation 3.4. Any point is covered by at most $m + 1$ objects of $\mathcal{A}$. If a point is covered by $m + 1$ objects of $\mathcal{A}$, it lies inside Q.

Similar to the 1.5D case, we now proceed to construct a terrain that has one convex vertex for each object of $\mathcal{A}$, and whose set of feasible vectors is exactly that object. The full details of the construction are given in [1].

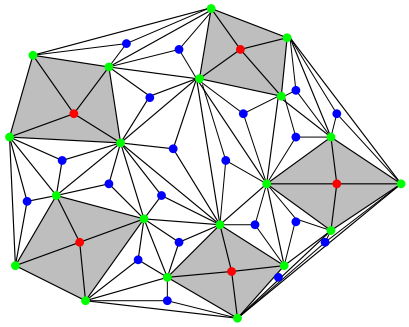


Figure 10 Projection of T onto the XY-plane.

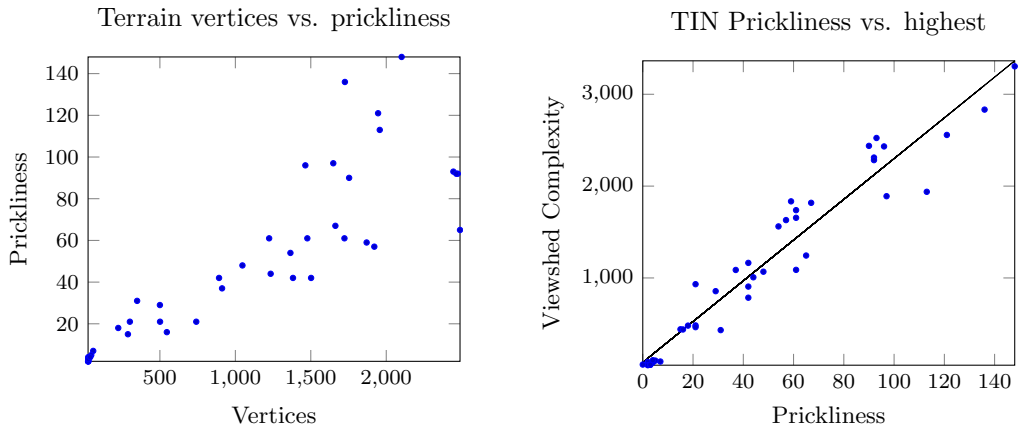


Figure 11 (left) The prickliness values for the terrains we considered. (right) The relation between the (median) viewshed complextity of the high viewpoints in a terrain and its prickliness.

We construct a terrain T with *red*, *green* and *blue* vertices; refer to Fig. 10. We assign one red vertex to Q and one to each of the rectangles intersecting it. The red vertices are placed at height two, and the distance between any pair of them is at least three. Each red vertex has four green vertices as neighbors, placed such that the extended cone of the red vertex intersected by the horizontal plane $z = 2$ is exactly the desired rectangle: the plane perpendicular to each red-green edge creates a line in the plane $z = 2$, and these lines can be chosen freely and independently. We then triangulate the resulting graph, and inside each completely green triangle we place a blue vertex u that is high enough so that (i) the three neighboring green vertices become non-convex vertices of T, and (ii) the cone of u intersected by the horizontal plane $z = 2$ contains Q. Then if the number of blue vertices is k, it is not difficult to see that the prickliness of T is $m + k + 1$ if and only if there is a point in Q covered by m rectangles. We obtain:

▶ **Theorem 3.5.** *The problem of computing the prickliness of a 2.5D terrain is 3SUM-hard.*

4 Experiments

We briefly report on some experiments relating the prickliness to the viewshed complexity in real world 2.5D terrains (see [12] for details). We considered 52 real-world terrains around the world. They varied in ruggedness, including mountainous regions (Rocky mountains,

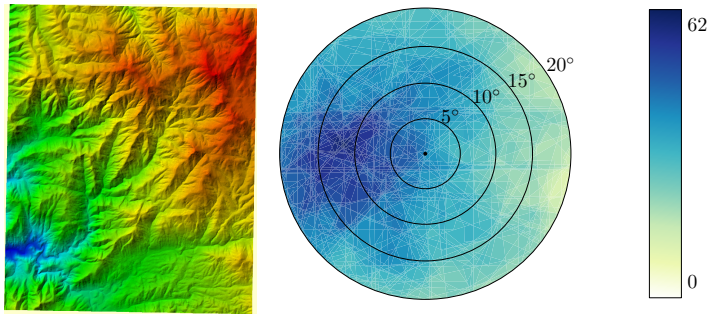


Figure 12 A real-world terrain with 583 vertices from the neighborhood of California Hot Springs whose prickliness is only 62. On the right the value $\pi_{\vec{v}}$ for vectors near $(0, 0, 1)$.

Himalaya), flat areas (farmlands in the Netherlands), and rolling hills (Sahara), and in number of vertices. For each of these terrains we computed the prickliness, and the viewsheds of nine "sufficiently separated" high points on the terrain. We refer to [12] on how we pick these points exactly. Fig. 11 (left) shows the number of vertices in each of the terrains and their prickliness. We see that the prickliness is generally much smaller than the number of vertices. See also Fig. 12. Fig. 11 (right) then shows the relation between the prickliness and the viewshed complexities. The viewshed complexities seem to scale nicely with the prickliness. This suggests that the prickliness may indeed be a valuable terrain descriptor.

Acknowledgments

The authors would like to thank Jeff Phillips for a stimulating discussion that, years later, led to the notion of prickliness.

A. A., R. J., and M. S. were supported by the Czech Science Foundation, grant number GJ19-06792Y, and by institutional support RVO: 67985807. M.L. was partially supported by the Netherlands Organization for Scientific Research (NWO) under project no. 614.001.504. R. S. was supported by projects MINECO PID2019-104129GB-I00 and Gen. Cat. DGR 2017SGR1640.

References

1 Ankush Acharyya, Ramesh K. Jallu, Maarten Löffler, Gert G.T. Meijer, Maria Saumell, Rodrigo I. Silveira, Frank Staals, and Hans Raj Tiwary. Terrain prickliness: theoretical grounds for low complexity viewsheds, 2021. arXiv:2103.06696 [cs.CG].

2 Maria Danese, Gabriele Nolè, and Beniamino Murgante. Identifying viewshed: New approaches to visual impact assessment. In *Geocomputation, Sustainability and Environmental Planning*, pages 73–89. Springer Berlin Heidelberg, 2011.

3 Mark de Berg, Herman J. Haverkort, and Constantinos P. Tsirogiannis. Visibility maps of realistic terrains have linear smoothed complexity. In *Proc. SoCG'09*, pages 163–168. ACM, 2009.

4 Herbert Edelsbrunner and Leonidas J Guibas. Topologically sweeping an arrangement. *J. Comput. System Sci.*, 38(1):165–194, 1989.

5 Anka Gajentaan and Mark H Overmars. On a class of $O(n^2)$ problems in computational geometry. *Comput. Geom.*, 5(3):165–185, 1995.

6 Ferran Hurtado, Maarten Löffler, Inês Matos, Vera Sacristán, Maria Saumell, Rodrigo I. Silveira, and Frank Staals. Terrain visibility with multiple viewpoints. *Internat. J. Comput. Geom. Appl.*, 24(04):275–306, 2014.

7 B. Joe and R. B. Simpson. Corrections to Lee's visibility polygon algorithm. *BIT*, 27(4):458–473, 1987.

8 Frank Kammer, Maarten Löffler, Paul Mutser, and Frank Staals. Practical approaches to partially guarding a polyhedral terrain. In *Proc. 8th International Conference on Geographic Information Science*, LNCS 8728, pages 318–332, 2014.

9 Young-Hoon Kim, Sanjay Rana, and Steve Wise. Exploring multiple viewshed analysis using terrain features and optimisation techniques. *Computers & Geosciences*, 30(9):1019–1032, 2004.

10 Anna Lubiw and András Rácz. A lower bound for the integer element distinctness problem. *Inf. Comput.*, 94(1):83–92, 1991.

11 Michael McKenna. Worst-case optimal hidden-surface removal. *ACM Trans. Graph.*, 6(1):19–28, 1987.

12 Gert Meijer. Realistic terrain features and the complexity of joint viewsheds. Master's thesis, 2020. URL: `https://dspace.library.uu.nl/handle/1874/399714`.

13 Esther Moet, Marc J. van Kreveld, and A. Frank van der Stappen. On realistic terrains. *Comput. Geom.*, 41(1-2):48–67, 2008.

14 Uta Schirpke, Erich Tasser, and Ulrike Tappeiner. Predicting scenic beauty of mountain regions. *Landscape Urban Plan.*, 111:1–12, 2013.

Tight Degree Bounds for Path-Greedy 5.19-Spanner on Convex Point Sets

William Evans[1] **and Lucca Siaudzionis**[2]

1 **University of British Columbia**
will@cs.ubc.ca

2 **University of British Columbia**
lsiau@cs.ubc.ca

Abstract

A t-spanner G for a set S of points in the plane is a weighted graph on the vertex set S, where the weight of an edge is the Euclidean distance between its endpoints, such that the shortest path between two points in G is at most t times their Euclidean distance. There has been an increasing interest in designing algorithms for constructing spanners of low degree. Bakhshesh and Fahri [1] asked whether the well-known path-greedy spanner algorithm can be used to find plane 5.19-spanners with maximum degree 3 on convex point sets. In this paper, we prove that path-greedy has a tight bound of 4 for the maximum degree of 5.19-spanners.

1 Introduction

Let S be a set of points in the plane. We say a graph G with vertex set S is *geometric* if each edge has weight equal to the Euclidean distance between its endpoints. For a real number $t > 1$, we say that geometric graph G is a *t-spanner* of S if the shortest path in G between each pair of vertices is at most t times the Euclidean distance between the vertices. A spanner is *plane* if the straight line segments between adjacent vertices of G do not cross each other, and the *degree* of a graph is the maximum degree of its vertices. There are several algorithms and problems related to geometric spanners [4, 6].

One classic example is the path-greedy spanner algorithm. It greedily builds a t-spanner from a point set S, using t as an input. Path-greedy is well known due to its simplicity. Algorithm 1 contains its implementation, and Figure 1 contains example outputs of path-greedy on two different convex point sets.

Algorithm 1: PathGreedy(S, t)

input : A set of S of N points in $\mathbb{R}^d$ and a real number $t > 1$.
output : $G(S, E)$, a t-spanner for S.

Sort all $\binom{n}{2}$ pairs of points from S in non-decreasing order of their distances, breaking ties arbitrarily, and store them in list L;
$E \leftarrow \emptyset$;
$G \leftarrow (S, E)$;
for each $\{p, q\} \in L$ **do**
  $\delta \leftarrow$ length of shortest path between p and q in G;
  if $\delta > t|pq|$ **then**
    $E \leftarrow E \cup \{(p, q)\}$;
return $G(S, E)$

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Figure 1 Two spanner graphs outputted by path-greedy on convex point sets, with $t = 2$.

1.1 Spanners of Low Degree

There has been a growing interest in plane spanners of low degree, since they can be used to design network topologies. Wireless networks can be modeled as geometric graphs, with their point set being related to the geographical location of the nodes. In that case, spanners can be used to design efficient communication amongst the nodes, and having those spanners be plane and of low degree is helpful for routing [5, 7].

Kanj et al. [5] proved that the lower bound for the maximum degree of t-spanners for points in convex position is 3. They also proved the bound is tight by providing an algorithm that finds a plane 20-spanner for convex point sets with maximum degree 3. Their algorithm guarantees maximum degree of 4 for general point sets. Biniaz et al. [3] designed another algorithm that constructs a plane $\left(\frac{3+4\pi}{3}\right)$-spanner for points in convex position, and Bakhshesh and Farshi [1] proposed a modified version of path-greedy that also guarantees a $\left(\frac{3+4\pi}{3}\right)$-spanner of maximum degree 3 for convex point sets.

1.2 Problem and Contribution

The work by Bakhshesh and Farshi lead to a natural question that they posed as an open problem: *Does* PathGreedy$(S, \frac{3+4\pi}{3})$ *construct a plane spanner of maximum degree 3 for points in convex position?*

The answer is no. Here, we show that the maximum degree of PathGreedy$(S, \frac{3+4\pi}{3})$ is 4, and that the bound is tight. In Section 2.1, we prove that for all convex point sets S and all real numbers $t \geq 4.27$, the algorithm PathGreedy(S, t) outputs a graph with maximum degree 4. In Section 2.2, we provide an example point set S^* such that the degree of PathGreedy$(S^*, \frac{3+4\pi}{3})$ is exactly 4.

2 A Tight Bound for the Maximum Degree of PathGreedy(S, 5.19)

In this section, we first prove that the maximum degree of a t-spanner for $t \geq 4.27$ generated by path-greedy for convex point sets is at most 4. Afterwards we show a convex point set S^* such that PathGreedy$(S^*, \frac{3+4\pi}{3})$ returns a graph with degree 4, alongside its construction. The construction can be generalized for all spanning ratios greater than or equal to 4.27.

We start with two technical lemmas, the first of which is an exercise in calculus.

▶ **Lemma 2.1.** *Given two angles α and β such that $\beta \geq \alpha > 0$ and $\pi > \alpha + \beta \geq 3\pi/4$, we have* $\cos\alpha + \cos\beta \leq \sqrt{2-\sqrt{2}}$ ◀

▶ **Lemma 2.2.** *Let $t \geq \frac{\sqrt{2}\cos(\pi/8)}{\sqrt{2}\cos(\pi/8)-1} = \frac{1}{1-\sqrt{2-\sqrt{2}}} \approx 4.262$. Given two angles α and β such that $\beta \geq \alpha > 0$ and $\pi > \alpha + \beta \geq 3\pi/4$:*

$$t\sin(\alpha+\beta) + \sin\alpha - t\sin\beta \leq 0$$

Proof. By the angle-sum identity for sin,

$$\begin{aligned} t\sin(\alpha+\beta)+\sin\alpha-t\sin\beta &= t\sin\alpha\cos\beta+t\sin\beta\cos\alpha+\sin\alpha-t\sin\beta \\ &= (t\cos\beta+1)\sin\alpha-(t-t\cos\alpha)\sin\beta \\ &= \left(t\cos\beta+1-\frac{\sin\beta}{\sin\alpha}t(1-\cos\alpha)\right)\sin\alpha \\ &\le (t\cos\beta+1-t(1-\cos\alpha))\sin\alpha \end{aligned}$$

($\frac{\sin\beta}{\sin\alpha} \ge 1$ since $\pi-\alpha>\beta\ge\alpha$)

$$\begin{aligned} &= (\cos\beta+\cos\alpha+1/t-1)t\sin\alpha \\ &\le \left(\sqrt{2-\sqrt{2}}+1/t-1\right)t \end{aligned}$$

(by Lemma 2.1 and $0<\sin\alpha<1$ since $0<\alpha<\pi/2$)

$$\le \left(\sqrt{2-\sqrt{2}}+1-\sqrt{2-\sqrt{2}}-1\right)t = 0$$

since $t \ge \frac{1}{1-\sqrt{2-\sqrt{2}}}$. ◀

2.1 The Upper Bound

▶ **Theorem 2.3.** *Given a convex point set S and a spanning ratio $t \ge 4.27$, the spanner graph generated by* PathGreedy*(S, t) has degree at most 4.*

Proof. Assume for a contradiction that vertex v has degree at least 5 in the spanner generated by PathGreedy(S, t). Let a, b, c, d, and e be five of the vertices that v is connected to. Assume without loss of generality that they are in clockwise order. Since the point set is convex, one of the angles $\angle avb$, $\angle bvc$, $\angle cvd$, and $\angle dve$ must be at most $\pi/4$. Assume without loss of generality that $\angle avb \le \pi/4$.

Let $\alpha = \angle vab$ and $\beta = \angle vba$. Assume without loss of generality that $|va| \ge |vb|$ and thus that $\beta \ge \alpha$. If $|va| = |vb|$, assume the edge (v, b) gets processed by path-greedy before edge (v, a). Since $\angle avb \le \pi/4$, we have $\beta \ge \alpha > 0$ and $\pi > \alpha + \beta \ge 3\pi/4$ and $|va| > |ab|$.

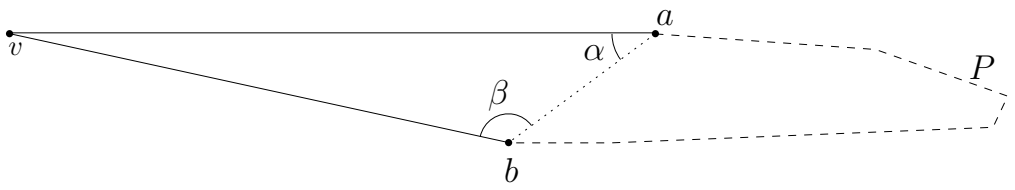


Figure 2 Illustration of Theorem 2.3's proof.

Note that, since $|va| > |ab|$, path-greedy will scan the edges (a, b) and (v, b) before the edge (v, a). However, it is not possible for path-greedy to add all three edges (a, b), (v, a), and (v, b), since $|ab| + |vb| < 2|va|$. Thus, since (v, a) and (v, b) are both added, path-greedy must skip the edge (a, b), which means that the vertices a and b must be connected by some path P such that:

$$|P| \le t|ab| \tag{1}$$

By the time path-greedy scans edge (v, a), the vertices v and a will be connected by the path $P + (v, b)$. Since path-greedy still adds the edge, this means that:

$$t|va| < |P| + |vb| \tag{2}$$

Combining inequalities (1) and (2):

$$t|ab| + |vb| - t|va| > 0. \tag{3}$$

The sine law for Δabv, $\frac{|va|}{\sin\beta} = \frac{|vb|}{\sin\alpha} = \frac{|ab|}{\sin(\alpha+\beta)}$, implies

$$|va| = \frac{\sin\beta}{\sin(\alpha+\beta)}|ab| \quad \text{and} \tag{4}$$

$$|vb| = \frac{\sin\alpha}{\sin(\alpha+\beta)}|ab| \tag{5}$$

Substituting (4) and (5) into (3):

$$\begin{aligned} t|ab| + \frac{\sin\alpha}{\sin(\alpha+\beta)}|ab| - t\frac{\sin\beta}{\sin(\alpha+\beta)}|ab| > 0 \quad &\Longrightarrow \\ t + \frac{\sin\alpha}{\sin(\alpha+\beta)} - t\frac{\sin\beta}{\sin(\alpha+\beta)} > 0 \quad &\Longrightarrow \\ t\sin(\alpha+\beta) + \sin\alpha - t\sin\beta > 0 \quad &(6) \end{aligned}$$

which contradicts Lemma 2.2. Thus, it is not possible for a vertex v to be connected to five or more vertices, making the degree of the spanner graph at most 4. ◀

2.2 The Lower Bound

▶ **Theorem 2.4.** *There exists a spanner generated by PathGreedy(S, $\frac{3+4\pi}{3}$) with degree 4.*

Proof. We prove by example. We constructed a convex point set S^* with no three collinear points such that the spanner generated by path-greedy has degree 4. The spanner is the same regardless of how ties are broken. The points in S^* are shown in Table 1.

Table 1 Coordinates and labels of point set S^*.

$a = (9.992, -0.399)$	$a_1 = (16.983, -0.749)$	$a_2 = (23.970, -1.169)$	$a_3 = (27.861, -2.099)$
$b = (5.258, -8.506)$	$b_1 = (12.253, -8.261)$	$b_2 = (19.246, -7.946)$	$b_3 = (26.235, -7.561)$
$v = (0, 0)$	$c = (-4.825, -8.758)$	$d = (-10, 0)$	

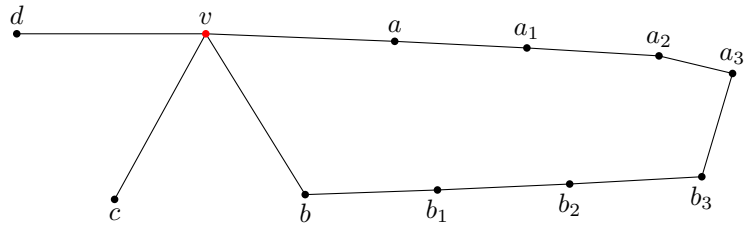


Figure 3 Spanner generated by PathGreedy(S^*, $\frac{3+4\pi}{3}$).

We can confirm that PathGreedy(S^*, $\frac{3+4\pi}{3}$) has degree 4 by running the algorithm. Figure 3 shows the spanner generated in this process. ◀

2.2.1 Construction of S^*

The points were not chosen at random, of course. First, point v was fixed at the origin. Then, points a, b, c, d were chosen such that $|va| \approx |vb| \approx |vc| \approx |vd| \approx 10$. Angles $\angle cvd$ and $\angle bvc$ are both slightly above $\pi/3$, and $\angle avb$ is slightly below $\pi/3$, such that $\angle dva < \pi$. To maintain convexity and non-collinearity, all the angles $\angle vaa_1$, $\angle aa_1a_2$, $\angle a_1a_2a_3$, $\angle b_3b_2b_1$, and $\angle b_2b_1b$ are below π.

The edges (a, a_1), (a_1, a_2), (a_2, a_3), (a_3, b_3), (b_3, b_2), (b_2, b_1), and (b_1, b) have length less than $|ab|$, so that the path P consisting of them is entirely added by path-greedy. It is crucial that $|P| \leq (\frac{3+4\pi}{3})|ab|$, so the edge (a, b) is not added, and $|P| + |vb| > (\frac{3+4\pi}{3})|va|$, so the edges (v, b) and (v, a) are both added. This is a scenario similar to the one explored in the proof of Theorem 2.3.

3 Conclusion

In this paper, we addressed the maximum degree of the path-greedy spanner algorithm's output for convex point sets. We proved that, for a convex point set S and a real number $t \geq 4.27$, the t-spanner generated by PathGreedy(S, t) will have its degree upper bounded by 4. Further, we showed an example point set S^* such that PathGreedy(S^*, $\frac{3+4\pi}{3}$) has degree 4, proving the bound for the maximum degree of path-greedy $\frac{3+4\pi}{3}$-spanner of 4 is tight. This solved part of an open problem suggested in [1].

Discovering whether the graph output of PathGreedy(S, $\frac{3+4\pi}{3}$) for convex point sets S is plane remains an open problem, which we intend to explore.

References

1 Davood Bakhshesh and Mohammad Farshi. A degree 3 plane 5.19-spanner for points in convex position. In *Proceedings of the 32nd Canadian Conference on Computational Geometry (CCCG2020)*, pages 226–232, 2020. URL: `https://vga.usask.ca/cccg2020/papers/Proceedings.pdf`.

2 Ahmad Biniaz, Mahdi Amani, Anil Maheshwari, Michiel H. M. Smid, Prosenjit Bose, and Jean-Lou De Carufel. A plane 1.88-spanner for points in convex position. *Journal of Computational Geometry*, 7(1):520–539, 2016.

3 Ahmad Biniaz, Prosenjit Bose, Jean-Lou De Carufel, Cyril Gavoille, Anil Maheshwari, and Michiel H. M. Smid. Towards plane spanners of degree 3. *Journal of Computational Geometry*, 8(1):11–31, 2017.

4 Prosenjit Bose and Michiel Smid. On plane geometric spanners: A survey and open problems. *Computational Geometry*, 46(7):818 – 830, 2013. EuroCG 2009. URL: `http://www.sciencedirect.com/science/article/pii/S0925772113000357`, `doi:https://doi.org/10.1016/j.comgeo.2013.04.002`.

5 Iyad Kanj, Ljubomir Perkovic, and Duru Turkoglu. Degree four plane spanners: Simpler and better. *Journal of Computational Geometry*, 8(2):3–31, 2017.

6 Giri Narasimhan and Michiel Smid. *Geometric spanner networks*. Cambridge University Press, Cambridge; New York, 2007.

7 Yu Wang and Xiang-Yang Li. Localized construction of bounded degree and planar spanner for wireless ad hoc networks. *Mobile Networks and Applications*, 11(2):161–175, 2006.

Crossing Numbers of Beyond-Planar Graphs Revisited

Nathan van Beusekom[1], Irene Parada[1], and Bettina Speckmann[1]

1 **TU Eindhoven, the Netherlands**
[n.a.c.v.beusekom|i.m.de.parada.munoz|b.speckmann]@tue.nl

1 Introduction

A central topic in graph drawing are high quality drawings which are not necessarily planar. A natural criterion for the quality of a drawing is the number of edge crossings. However, it is NP-hard to compute the minimum number of crossings of a graph G across all possible drawings of G, also referred to as the *crossing number* of G, $\mathrm{cr}(G)$ [7]. Meaningful upper and lower bounds, as well as heuristic, approximation, and parameterized algorithms, have hence been a major focus [3].

Drawing edges as straight-line segments arguably simplifies any drawing. One classic variant is hence the *rectilinear crossing number* $\overline{\mathrm{cr}}(G)$ of a graph G, that is, the minimum number of crossings across all straight-line drawings of G. Clearly $\mathrm{cr}(G) \leq \overline{\mathrm{cr}}(G)$ for any graph G. The straight-line restriction can increase the crossing number arbitrarily: Bienstock and Dean [2] showed that for every k there exists a graph G such that $\mathrm{cr}(G) = 4$ and $\overline{\mathrm{cr}}(G) = k$.

In recent years there has been particular interest in drawings of *beyond-planar* graphs, which are characterized by certain forbidden crossing configurations of the edges; see the recent survey by Didimo *et al.* [6]. In this paper we study the relation between the crossing number restricted to beyond-planar drawings, and the (unrestricted) crossing number. The *restricted crossing number* $\mathrm{cr}_{\mathcal{F}}(G)$ of a graph G is the minimum number of crossings required to draw G such that the drawing belongs to the beyond-planar family $\mathcal{F}$. For a beyond-planar family $\mathcal{F}$, the *crossing ratio* $\varrho_{\mathcal{F}}$ is defined as $\varrho_{\mathcal{F}} = \sup_{G \in \mathcal{F}} \mathrm{cr}_{\mathcal{F}}(G)/\mathrm{cr}(G)$, that is, the supremum over all graphs in $\mathcal{F}$ of the ratio between the restricted crossing number and the (unrestricted) crossing number.

Results Chimani *et al.* [5] gave bounds on the crossing ratio for the 1-planar, quasi-planar, and fan-planar families. These results show that there exist graphs that have significantly larger crossing numbers when drawn with the beyond-planar restrictions. Their 1-planarity bound also applies to k-planarity when allowing parallel edges. We show that parallel edges are not needed when extending their proof to k-planarity. To do so, we introduce the concept of *k-planar compound edges*, which exhibit essentially the same behavior as k parallel edges. See the full paper for details. In Section 2 and the full paper we show how to extend their proof constructions to four additional classes of beyond-planar graphs. In Section 3 we introduce the concept of *ℓ-bundles* which allow us to prove tight bounds on the crossing ratio for families of graphs where these bounds are quadratic in the number n of vertices.

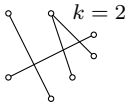
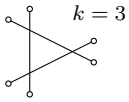
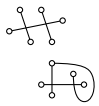
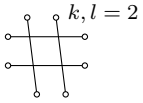
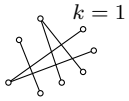
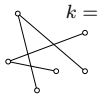
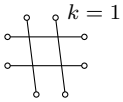
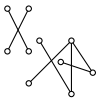
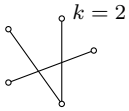
All bounds are summarized in Table 1. Bounds that are tight for a fixed k are indicated in boldface. In the full paper we show that all bounds also apply to straight-line drawings.

2 (k, l)-Grid-Free and k-Gap-Planar

Chimani *et al.* [5] use the notion of extended edges, which we refer to as *ℓ-compound edges*: an ℓ-compound edge xy is a combination of the edge xy and a set Π_{xy} of $\ell - 1$ paths of length two all

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Table 1 Bounds for the supremum ϱ of the ratio between crossing numbers, for our upper bounds we assume the drawings to be *simple*, that is, two edges share at most one point.

Family	Forbidden Configurations		Lower	Upper
k-planar	An edge crossed more than k times		$\boldsymbol{\Omega(n/k)}$ (full paper)	$\boldsymbol{O(k\sqrt{k}n)}$ (full paper)
k-quasi-planar	k pairwise crossing edges		$\Omega(n/k^3)$ [5]	$f(k)n^2\log^2 n$ [5]
Fan-planar	Two independent edges crossing a third or two adjacent edges crossing another edge from different "sides"		$\Omega(n)$ [5]	$O(n^2)$ [5]
(k,l)-grid-free	Set of k edges such that each edge crosses each edge from a set of l edges.		$\Omega\left(\frac{n}{kl(k+l)}\right)$ (Section 2.1)	$g(k,l)n^2$ (Section 2.1)
k-gap-planar	More than k crossings mapped to an edge in an optimal mapping		$\boldsymbol{\Omega(n/k^3)}$ (Section 2.2)	$\boldsymbol{O(k\sqrt{k}n)}$ (Section 2.2)
Skewness-k	Set of crossings not covered by at most k edges		$\boldsymbol{\Omega(n/k)}$ (full paper)	$\boldsymbol{O(kn+k^2)}$ (full paper)
k-apex	Set of crossings not covered by at most k vertices		$\Omega(n/k)$ (full paper)	$O(k^2n^2+k^4)$ (full paper)
Planarly connected	Two crossing edges that do not have two of their endpoint connected by a crossing-free edge		$\boldsymbol{\Omega(n^2)}$ (Section 3.1)	$\boldsymbol{O(n^2)}$ (Section 3.1)
k-fan-crossing-free	An edge that crosses k adjacent edges		$\boldsymbol{\Omega(n^2/k^3)}$ (Section 3.2)	$\boldsymbol{O(k^2n^2)}$ (Section 3.2)
Straight-line RAC	Two edges crossing at an angle $< \frac{\pi}{2}$		$\boldsymbol{\Omega(n^2)}$ (full paper)	$\boldsymbol{O(n^2)}$ (full paper)

connecting x and y (see Figure 1). We indicate ℓ-compound edges with thick blue lines. In their Lemma 5, Chimani *et al.* show that if two ℓ-compound edges cross in a drawing Γ, then Γ contains at least ℓ crossings. They then use closed curves to separate the endpoints of an ℓ-compound edge to prove that a drawing contains at least ℓ crossings. They use this technique to prove bounds on the crossing ratio for k-quasi-planarity and fan-planarity and conjectured that similar constructions could be used on additional families of beyond-planar graphs. Using similar proof structures, in this section we extend their results to two more classes

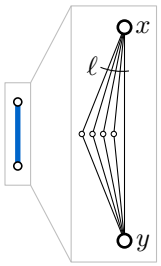


Figure 1 An ℓ-compound edge.

of beyond-planar graphs: (k, l)-grid-free graphs and k-gap-planar graphs. For both of the classes we achieve lower bounds on the crossing ratio which are linear in the number of vertices n. For the (k, l)-grid-free crossing ratio we provide a general bound for all $k > 1$ and $l > 1$. For the k-gap-planar crossing ratio, we first provide a tight lower bound for 1-gap-planarity and then show how to extend it to a bound for k-gap-planarity.

2.1 (k, l)-Grid-Free Graphs

In a (k, l)-grid-free drawing it is forbidden to have a $k \times l$ grid, that is, a set of k edges that cross a set of l edges. We show that $\varrho_{k,l-\text{grid}-\text{free}} \in \Omega\Big(\frac{n}{kl(k+l)}\Big)$ by creating a graph where a (k, l)-grid occurs when we draw all grid edges (drawn in black) in the same face of a cycle of ℓ-compound edges (see Figure 2). By Lemma 5 of [5] if two edges in the cycle cross there would be at least ℓ crossings. In an (k, l)-grid-free drawing, one grid edge has to be drawn in the same face (with respect to the cycle) as the vertex x and must hence cross an ℓ-compound edge (drawn in blue), incurring ℓ crossings.

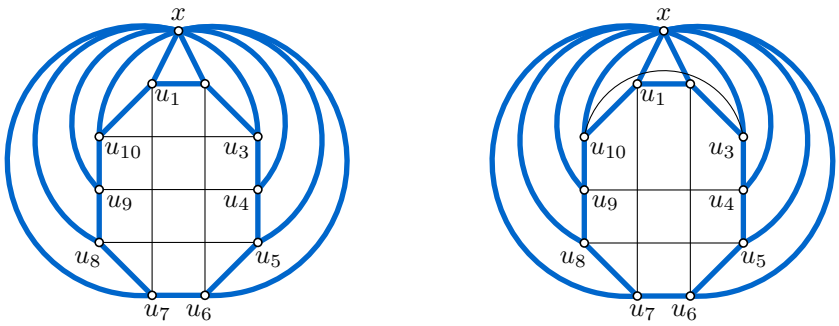


Figure 2 Left: drawing of G_ℓ with $\text{cr}(G_\ell) \leq 6$; Right: $(2, 3)$-grid-free drawing of G_ℓ with $\text{cr}_{2,3-\text{grid}-\text{free}}(G_\ell) \geq \ell$, ℓ-compound edges drawn with thick blue lines.

2.2 k-Gap-Planar Graphs

In a k-gap-planar drawing every crossing is assigned to one of the two edges involved. No more than k crossings can be assigned to a single edge. We first describe our lower bound construction for $k = 1$ and then show how to extend it to larger k. As before we create a cycle C of length eight of ℓ-compound edges and connect each vertex of the cycle to an additional vertex x using a further eight ℓ-compound edges. Again by Lemma 5 of [5] none of the ℓ-compound edges can cross. We add four single edges which connect diametrically opposite vertices of C. See Figure 3 (left): ℓ-compound edges are indicated in blue, single edges in black. In a 1-gap-planar drawing one single edge has to be drawn in the same face (with respect to the cycle) as the vertex x and must hence cross an ℓ-compound edge, incurring ℓ crossings.

To prove a lower bound for k-gap-planarity, we can use exactly the same construction as above starting with a cycle of length $8k$. Our lower bound $\varrho_{k-\text{gap}} \in \Omega(n/k^3)$ is cubic in $1/k$ and hence in general does not match the upper bound.

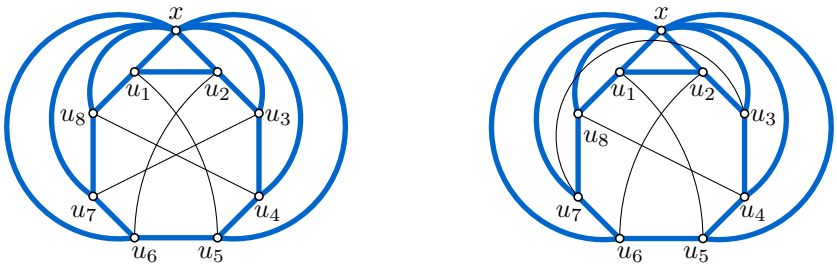


Figure 3 Left: drawing of G_ℓ with $\mathrm{cr}(G_\ell) \leq 6$; Right: 1-gap-planar drawing of G_ℓ with $\mathrm{cr}_{1-\mathrm{gap}}(G_\ell) \geq \ell$, ℓ-compound edges are drawn with thick blue lines.

3 Planarly Connected Graphs and k-Fan-Crossing-Free Graphs

In this section we introduce the concept of ℓ_2-*bundles* and ℓ_3-*bundles*, which are sets of ℓ paths of length two or three, respectively, all connecting the same two vertices (see Figure 4). We indicate ℓ_2-bundles and ℓ_3-bundles with thick red and green lines, respectively. The vertices that are interior to a bundle have degree two. Two crossing bundles result in a quadratic number of crossings. We show how to enforce such crossings to prove tight bounds on the crossing ratio. Lemma 1, proven in the full paper, argues that there is a planarly connected drawing that is crossing minimal in which ℓ_2-bundles do not cross themselves and no vertex lies between two consecutive paths of the same ℓ_2-bundle.

Figure 4 Left: an ℓ_2-bundle; Right: an ℓ_3-bundle; drawn as a thick red or green line, respecitvely.

3.1 Planarly Connected Graphs

In a planarly connected drawing two edges can cross only if their endpoints are connected by another edge which has no crossings. We show that $\varrho_{\mathrm{pl-con}} \in \Omega(n^2)$ using ℓ_2-bundles.

▶ **Lemma 1.** *If a planarly connected graph G contains ℓ_2-bundles, then there is a planarly connected crossing-minimal drawing Γ of G in which paths from the same ℓ_2-bundle do not cross each other, all paths of the same ℓ_2-bundle cross the same edges, and no vertex lies between two consecutive paths of the same ℓ_2-bundle.*

▶ **Theorem 2.** *For every $\ell \geq 2$ there exists a planarly connected graph G_ℓ with $n = 7\ell + 6$ vertices such that $\mathrm{cr}_{\mathrm{pl-con}}(G_\ell) \geq \ell^2$ and $\mathrm{cr}(G_\ell) \leq 1$. Thus, $\varrho_{\mathrm{pl-con}} \in \Theta(n^2)$.*

Proof. The upper bound $\varrho_{\mathrm{pl-con}} \in O(n^2)$ follows directly from the fact that a planarly connected graph with n vertices has at most cn edges, where c is a constant [1].

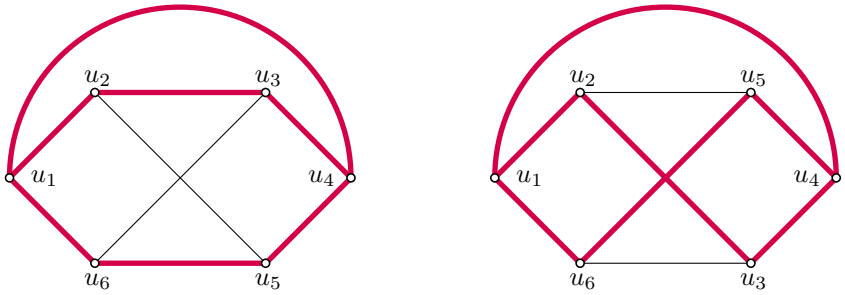


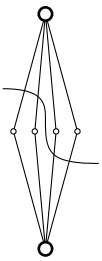
Figure 5 Left: drawing of G_ℓ with $\mathrm{cr}(G_\ell) = 1$; Right: planarly connected drawing of G_ℓ with $\mathrm{cr}_{\mathrm{pl-con}}(G_\ell) = \ell^2$, ℓ_2-bundles are drawn with thick red lines.

For the lower bound, we construct a graph G_ℓ. Start with an 6-cycle $\langle u_1, u_2, \ldots, u_6\rangle$, and an edge u_1u_4, resulting in the graph G'. Replace each edge in G' with an ℓ_2-bundle. The graph G_ℓ has $n = 7\ell + 6$ vertices and admits a drawing with only one crossing (see Figure 5, left).

By Lemma 1 there is a planarly connected crossing-minimal drawing Γ of G_ℓ in which for every ℓ_2-bundle its edges do not cross each other, all paths cross the same edges, and no vertex lies between two consecutive paths. Let C be the cycle of ℓ_2-bundles $C = \langle u_1, u_4, u_5, u_6\rangle$. If C crosses itself in Γ, then there are ℓ^2 crossings. Otherwise C is drawn without crossings in Γ and defines two faces of size 8. The other faces do not contain vertices of Γ in their interior. Thus, u_2 and u_3 can either be drawn in the same face defined by C, or in different faces. If they are drawn in the same face, Γ cannot be planarly connected: a simple case distinction shows that there has to be a crossing between two edges whose endpoints are not connected by any other edge. Specifically, there is a crossing between either u_1u_2 and u_3u_4, u_1u_2 and u_3u_6, u_2u_5 and u_3u_6 or u_2u_5 and u_3u_4. These are all not allowed, as each time there is no edge connecting the two edges. If u_2 and u_3 are drawn in different faces defined by C, then the ℓ_2-bundle u_2u_3 has to cross C. As C is composed of ℓ_2-bundles, there are ℓ cycles, distinct in edges, all separating u_3 and u_4. All the ℓ distinct paths in the ℓ_2-bundle u_2u_3 have to cross the ℓ distinct (in edges) cycles in C. Therefore, Γ has at least ℓ^2 crossings. ◀

3.2 k-Fan-Crossing-Free Graphs

In a k-fan-crossing-free drawing it is forbidden for an edge to cross k edges which are adjacent to the same vertex. While it is possible for an edge to "weave" through an ℓ_2-bundle with $\ell \leq 2k - 2$ (see right), no edge can cross an ℓ_2-bundle with $\ell \geq 2k - 1$. In contrast, ℓ_3 bundles can be crossed for any ℓ. We show that $\varrho_{\mathrm{fan-free}} \in \Omega(n^2)$ and $\varrho_{k-\mathrm{fan-free}} \in \Omega(n^2/k^3)$ using ℓ_2-bundles to force ℓ_3-bundles to cross.



▶ **Lemma 3.** *Let G be a k-fan-crossing-free graph and let p be the length of the longest simple path in G. If G contains ℓ_2-bundles with $\ell \geq 2p(k-1)+1$, then in any crossing-minimal k-fan-crossing-free drawing of G the edges in the ℓ_2-bundles have no crossings.*

▶ **Theorem 4.** *For every $\ell \geq 95$ there exists a fan-crossing-free graph G_ℓ with $n = 75\ell + 16$ vertices such that $\mathrm{cr}_{\mathrm{fan-free}}(G_\ell) \geq \ell^2/2$ and $\mathrm{cr}(G_\ell) \leq 2$. Thus, $\varrho_{\mathrm{fan-free}} \in \Theta(n^2)$.*

Proof. The upper bound $\varrho_{\mathrm{pl-con}} \in O(n^2)$ follows directly from the fact that a fan-crossing-free graph with n vertices can have at most $4n - 8$ edges [4].

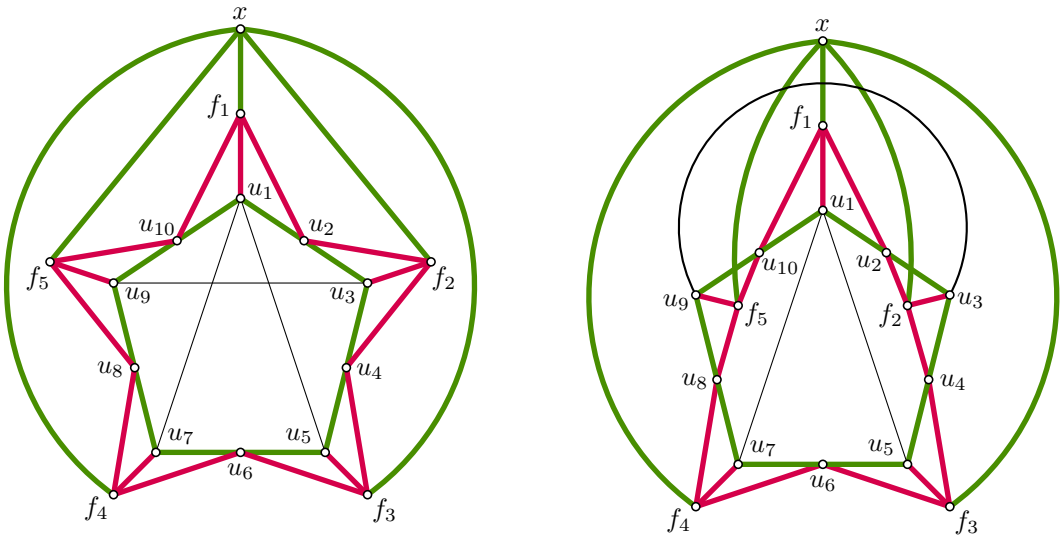


Figure 6 Left: drawing of G_ℓ with $\mathrm{cr}(G_\ell) \leq 2$; Right: fan-crossing-free drawing of G_ℓ with $\mathrm{cr}_{\mathrm{fan-free}}(G_\ell) \geq \ell^2$. Thick thick red (green) lines represent ℓ_2-bundles (ℓ_3-bundles).

For the lower bound we construct a graph G_ℓ, see Figure 6. We start with a 10-cycle $C = \langle u_1, u_2 \ldots, u_{10} \rangle$. We add five vertices $f_1, f_2, \ldots, f_5$ and add the edges $f_i u_{2i-2}, f_i u_{2i-1}, f_i u_{2i}$ for $i = 1, 2, \ldots, 5$. We add a new vertex x and connect it to each f_i for $i = 1, 2, \ldots, 5$. We then replace each edge $f_i u_j$ by an ℓ_2-bundle and replace each edge $x f_i$ and $u_i u_j$ by an ℓ_3-bundle. Finally, we add the *regular* edges $u_1 u_5, u_1 u_7$, and $u_3 u_9$. The resulting graph G_ℓ has $n = 75\ell + 16$ vertices and admits a drawing with only two crossings (see Figure 6, left).

In G_ℓ there is a cycle $F = \langle f_1, u_2, f_2, u_4, f_3, u_6, f_4, u_8, f_5, u_{10} \rangle$ of ℓ_2-bundles. There are $10+5+1 = 16$ vertices which are not interior to bundles. The longest path can use at most two vertices inside each bundle and has length at most $48-1 = 47$. Since $\ell \geq 95 = 2 \cdot 47 \cdot (2-1)+1$, by Lemma 3, all ℓ_2-bundles must be drawn without crossings and F must be drawn plane. If u_1, u_3, u_5, u_7, and u_9 all lie in the same face of F then we are effectively in the situation depicted in Figure 6 (left) and the drawing is not fan-crossing-free. Hence u_1, u_3, u_5, u_7, and u_9 must distribute over the two faces of F. Specifically, we can assume that there a vertex u_i, $i \in \{1, 3, 5, 7, 9\}$, such that u_i lies in the same face of F as x.

We consider the cycle Z consisting of the two ℓ_2-bundles between u_i and $f_{(i+1)/2}$ and between $f_{(i+1)/2}$ and u_{i-1}, and the ℓ_3-bundle between u_i and u_{i-1}. Symmetrically, we can consider the cycle Z' involving u_{i+1} instead of u_{i-1}. The vertex x can lie within any face of the ℓ_3-bundle. Consider the cycles formed by each path of the ℓ_3-bundle together with the two ℓ_2-bundles. These cycles divide the plan into two faces. We say that the *exterior* face of each cycle is the face that contains F. We say that x lies in the *exterior* of Z (Z', respectively), if it lies in the exterior face of $\ell/2$ of these cycles. Otherwise, x lies in the *interior* of Z (Z', respectively). Observe that x lies either in the interior of Z, or in the interior of Z', or in the exterior of both.

If x lies in the exterior face of both Z and Z', then the ℓ_3-bundle connecting x and $f_{(i+1)/2}$ crosses at least $\ell/2$ paths in either Z or Z' (see Figure 6, right, with $u_i = u_3$ and $f_{(i+1)/2} = f_2$). If x lies in the interior face of Z, then (by definition of the faces of Z) there is an $f_j \neq f_{(i+1)/2}$ which lies in the exterior face of Z. The ℓ_3-bundle connecting x and f_j crosses at least $\ell/2$ paths in Z. The argument for Z' is symmetric. ◀

The proof can be extended to k-fan-crossing-free graphs by creating a cycle C of length $6 + 2k$ which results in a graph with $3/2 \cdot (6 + 2k) + 1 = 10 + 3k$ vertices not interior to bundles and a longest path of length at most $29 + 9k$. Since no path should be able to cross the ℓ_2-bundles, we require that $\ell \geq 2 \cdot (29 + 9k) \cdot (k - 1) + 1 = 18k^2 + 40k - 57$.

▶ **Corollary 5.** *For every $\ell \geq 18k^2 + 40k - 57$ there exists a k-fan-crossing-free graph G_ℓ^k with $n = 15k\ell + 45\ell + 3k + 10$ vertices such that $\mathrm{cr}_{k-\mathrm{fan}-\mathrm{free}}(G_\ell^k) \geq \ell^2$ and $\mathrm{cr}(G_\ell^k) \leq k$. Thus, $\varrho_{k-\mathrm{fan}-\mathrm{free}} \in \Omega(n^2/k^3)$.*

References

1 Eyal Ackerman, Balázs Keszegh, and Mate Vizer. On the size of planarly connected crossing graphs. *Journal of Graph Algorithms and Applications*, 22(1):11–22, 2018. doi:10.7155/jgaa.00453.

2 Daniel Bienstock and Nathaniel Dean. Bounds for rectilinear crossing numbers. *Journal of Graph Theory*, 17(3):333–348, 1993. doi:10.1002/jgt.3190170308.

3 Christoph Buchheim, Markus Chimani, Carsten Gutwenger, Michael Jünger, and Petra Mutzel. Crossings and planarization. In Roberto Tamassia, editor, *Handbook on Graph Drawing and Visualization*, pages 43–85. CRC Press, 2013. doi:10.1201/b15385-5.

4 Otfried Cheong, Sariel Har-Peled, Heuna Kim, and Hyo-Sil Kim. On the number of edges of fan-crossing free graphs. *Algorithmica*, 73(4):673–695, 2014. doi:10.1007/s00453-014-9935-z.

5 Markus Chimani, Philipp Kindermann, Fabrizio Montecchiani, and Pavel Valtr. Crossing numbers of beyond-planar graphs. In *Proceedings of the 27th International Symposium on Graph Drawing (GD 2019)*, LNCS 11904, pages 78–86, 2019. doi:10.1007/978-3-030-35802-0_6.

6 Walter Didimo, Giuseppe Liotta, and Fabrizio Montecchiani. A survey on graph drawing beyond planarity. *ACM Computing Surveys*, 52(1):1–37, 2019. doi:10.1145/3301281.

7 Michael Garey and David S. Johnson. Crossing number is NP-complete. *SIAM Journal on Algebraic and Discrete Methods*, 4(3):312–316, 1983. doi:10.1137/0604033.

Reducing Moser's Square Packing Problem to a Bounded Number of Squares

Meike Neuwohner[1]

1 **Research Institute for Discrete Mathematics, Universität Bonn**
neuwohner@or.uni-bonn.de

Abstract

The problem widely known as *Moser's Square Packing Problem* asks for the smallest area A such that for any set S of squares of total area 1, there exists a rectangle R of area A into which the squares in S permit an internally-disjoint, axis-parallel packing. It was formulated by Moser [7] in 1966 and remains unsolved so far. The best known lower bound of $\frac{2+\sqrt{3}}{3} \leq A$ is due to Novotný [8] and has been shown to be sufficient for up to 11 squares by Platz [11], while Hougardy [1] and Ilhan [2] have established that $A < 1.37$.
In this paper, we reduce Moser's Square Packing Problem to a problem on a finite set of squares in the following sense: We show how to compute a natural number N such that it is enough to determine the value of A for sets containing at most N squares with total area 1.

Related Version arXiv:2103.06597

1 Introduction

In 1966, Leo Moser [7] asked for the minimum area A such that for any (possibly infinite) set S of squares of total area 1, there exists a rectangle R of area A and an axis-parallel, internally disjoint packing of S into R. The first bounds of $1.2 < A \leq 2$ were provided by Moon and Moser [6] in 1967. Their idea was generalized by Meir and Moser [5] to prove the following theorem:

▶ **Theorem 1.** *For $k \in \mathbb{N}_{>0}$, any (possibly infinite) set of k-dimensional hypercubes with edge lengths given by $x_1 \geq x_2 \geq \ldots$ and finite total volume V permits an axis-parallel, internally disjoint packing into any rectangular parallelepiped with edge lengths $a_1, \ldots, a_k$ satisfying $a_j \geq x_1$ for all $j = 1 \ldots, k$ as well as $x_1^k + \Pi_{j=1}^k (a_j - x_1) \geq V$.*

The special case $k = 2$ has found several applications in subsequent works [9], [1].
In 1970, Kleitman and Krieger [3] proved an upper bound of $A \leq \sqrt{3} < 1.733$, which they could improve to $\frac{4}{\sqrt{6}} < 1.633$ for *finite* sets of squares [4]. Their argument was generalized to the case of possibly infinite families of squares by Zernisch [12] in 2012.
The best known lower bound of $\frac{2+\sqrt{3}}{3}$, which is actually believed to be the correct answer to the problem, was provided by Novotný [8] in 1995 (see Figure 1). For packing up to 5 squares, he proved that an area of $\frac{2+\sqrt{3}}{3}$ indeed suffices [10], which was extended to up to 11 squares by Platz [11]. Moreover, using the results from [5] and [4], Novotný [9] proved an upper bound of $A < 1.53$ for the general case of possibly infinitely many squares.
The best known upper bound of $A < 1.37$ is due to Hougardy [1] and Ilhan [2].
In this paper, we present a general approach that allows us to reduce the task of showing that any set of squares of total area 1 can be packed into a rectangle of area $\frac{2+\sqrt{3}}{3} \leq F \leq 1.37$ to the problem of proving this statement for *finite* sets containing at most N squares, where N is a natural number that can be easily computed.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

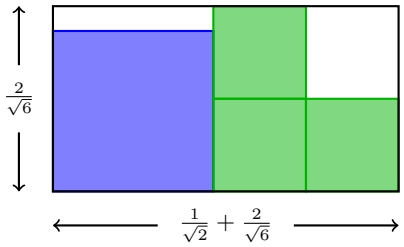


Figure 1 The lower bound example provided by Novotný.

2 Reduction to a Finite Number of Squares

Let a target area $F \geq \frac{2+\sqrt{3}}{3}$ be given. This section shows how to compute a natural number $N(F)$ with the following property: If for any set of at most $N(F)$ squares of total area 1, there exists a rectangle of area F into which these squares can be packed axis-parallel and internally disjoint, then the same statement holds for any, possibly infinite set of squares of total area 1.

From the way we compute $N(F)$, it is not hard to see that $N := \max\{N(F), F \in [\frac{2+\sqrt{3}}{3}, 1.37]\}$ exists and is attained by $N(\frac{2+\sqrt{3}}{3}) > 4$. Note that these properties already imply that if we denote the correct answer to Moser's Square Packing Problem (when restricted to finite sets containing at most N squares) by A (A'), then $A = A'$.

In order to determine numbers $N(F), F \in [\frac{2+\sqrt{3}}{3}, 1.37]$ as promised, we first note that without loss of generality, we can assume a set of squares of total area 1 to be given by a non-increasing sequence $s_1 \geq s_2 \geq \ldots$ of non-negative edge lengths satisfying $\sum_{i=1}^{\infty} s_i^2 = 1$. This is because for any set S of squares of total area 1, the number of squares in S with edge lengths in $[\frac{1}{n+1}, \frac{1}{n}]$ for some fixed positive integer n must be finite, and squares of edge length zero do not influence the packing task. For a given sequence $s_1 \geq s_2 \geq \ldots$, we denote the corresponding squares by $S_1, S_2, \ldots$.

We further observe that if s_1 is small enough, the criterion established by Meir and Moser provides a packing of S into a rectangle of area F, so we can assume that s_1 is greater than a certain constant. Next, we show that there exists constants $N_0(F)$ and $c(F)$ with the following properties: If the total area of the squares with indices $N_0(F)+1$, $N_0(F)+2$, ... amounts to more than $c(F)^2$, (implying that the total area of the first $N_0(F)$ largest squares is less than $1 - c^2(F)$,) then their edge lengths must be "small" and they can, therefore, be packed "efficiently". On the other hand, if the total area of $S_{N_0(F)+1}$, $S_{N_0(F)+2}$, ... is at most $c(F)^2$, then we also know that there must be some $N_0(F)+1 \leq n \leq \lfloor e^2 \cdot N_0(F) \rfloor$ such that $s_n \leq \frac{c}{\sqrt{n}}$. For such n, we show that for any packing of the first n squares into a rectangle of area F, the remaining squares can be accommodated in the arising *whitespace* (see Definition 2). This implies that it suffices to consider sets of at most $N(F) := \lfloor e^2 \cdot N_0(F) \rfloor$ squares to find the correct answer to Moser's Square Packing Problem. The remainder of this section provides the omitted details. Let $F \geq \frac{2+\sqrt{3}}{3}$ be fixed.

▶ **Definition 2** (whitespace). Let a closed rectangle R and a set $\mathcal{S}$ of closed squares be given, such that each of the squares in $\mathcal{S}$ is contained within R and the interiors of two distinct squares from $\mathcal{S}$ do not intersect. We define the *whitespace* whitespace$(R, \mathcal{S})$ arising from this packing of the squares in $\mathcal{S}$ into R to equal whitespace$(R, \mathcal{S}) := \overline{R \setminus \bigcup_{S \in \mathcal{S}} S}$, where for a set A of points in the Euclidean plane, $\bar{A}$ denotes its topological closure.

▶ **Lemma 3.** *Let* $c := \sqrt{\left(\frac{3}{10}\right)^2 + \frac{F-1}{5}} - \frac{3}{10}$. *Then for* $n \geq \max\{(10F + \frac{1}{10})^2, 100c^2\}$ *and any*

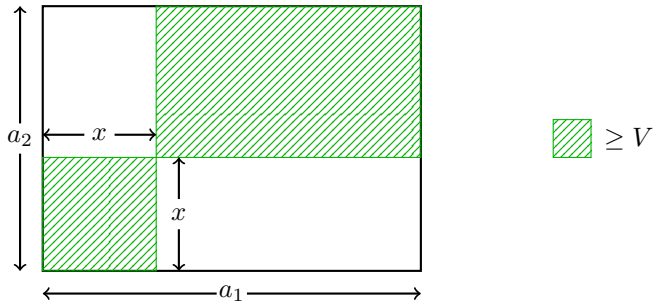


Figure 2 Visualization of the condition established by Meir and Moser for $k = 2$. The picture does *not* indicate the packing strategy applied to prove the respective result.

packing of n squares of total area at most 1 into a rectangle with edge lengths $\frac{1}{10} \leq W \leq H$ and area $W \cdot H = F$, any set of squares of total area at most c^2 and maximum edge length at most $\frac{c}{\sqrt{n}}$ permits an axis-parallel, internally disjoint packing into the arising whitespace.

Proof. Let $n \geq \max\{(10F + \frac{1}{10})^2, 100c^2\}$ be a natural number and let n squares $S_1, \ldots, S_n$ with edge lengths $s_1, \ldots, s_n$ and total area at most 1 be given. Consider a packing of these squares into an enclosing rectangle R with edge lengths $\frac{1}{10} \leq W \leq H$ and area F.
Moreover, let a set of squares of total area at most c^2 and maximum edge length at most $\frac{c}{\sqrt{n}}$ be given by a sequence $\frac{c}{\sqrt{n}} \geq s_{n+1} \geq s_{n+2} \geq \ldots$. For a finite set, we have $s_j = 0$ from some point on.
Let $k \geq n + 1$ and assume that squares of edge lengths $s_{n+1}, \ldots, s_{k-1}$ have already been accommodated within the whitespace. Our goal is to find a lower bound on the area of the set of points in R that serve as potential midpoints for squares of edge length s_k contained in the remaining whitespace. If we can provide a positive lower bound on the area of the set of potential square midpoints for each $k \geq n + 1$ for which $s_k > 0$, this in particular implies that this set will never be empty. Hence, we can inductively construct a packing of the squares with edge length $s_{n+1} \geq s_{n+2} \geq \ldots$ into the whitespace by always placing the next square at the lexicographically minimum feasible position. The latter exists as the space of feasible positions is compact.
Let $k \geq n+1$ such that $s_k > 0$ and assume that squares $(S_i)_{i=n+1}^{k-1}$ with edge lengths $(s_i)_{i=n+1}^{k-1}$ have been placed axis-parallel and internally disjoint within the whitespace. Observe that the square of edge length s_k with midpoint $p \in R$ is contained within the (remaining) whitespace if and only if the l_∞-distance of p to each of the S_i for $i = 1, \ldots, k-1$ and to the boundary of R is at least $\frac{s_k}{2}$. For $i = 1, \ldots, k-1$, let S_i' denote the square arising from S_i by extending it by $\frac{s_k}{2}$ in positive and negative x- and y-direction and let $F_i := S_i' \backslash S_i$ for $i = 1, \ldots, n$. Moreover, let R' be the rectangle with edge lengths $W - s_k$ an $H - s_k$ lying centered within R (note that $s_k \leq \frac{c}{\sqrt{n}} \leq \frac{1}{10} \leq W \leq H$ by our choice of n) and let further $F_R := R \backslash R'$. Then, for a point p openly contained in the original whitespace, the square with midpoint p and edge length s_k is contained in the remaining whitespace after additionally packing $(S_i)_{i=n+1}^{k-1}$ if and only if p is not openly contained within any of the frames $(F_i)_{i=1}^n$ around the squares $(S_i)_{i=1}^n$, any of the squares $(S_i')_{i=n+1}^{k-1}$, or the frame F_R inscribed into R. Figure 3 illustrates the case $k = n + 1$. In particular, as the area of the (original) whitespace is at least $F - 1$, the area of the set M of potential midpoints of

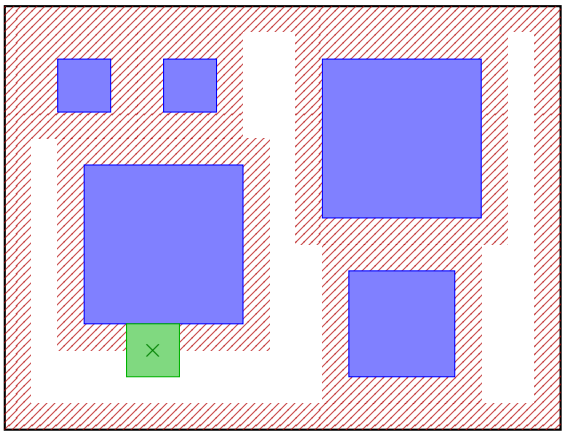


Figure 3 The white area that is not shaded remains feasible for midpoints of whitespace squares.

whitespace squares amounts to

$$\operatorname{area}(M) \geq F - 1 - \sum_{i=1}^{n} \operatorname{area}(F_i) - \operatorname{area}(F_R) - \sum_{i=n+1}^{k-1} \operatorname{area}(S'_i).$$

$$\text{We have} \quad \operatorname{area}(F_i) = 4 \cdot s_i \cdot \frac{s_k}{2} + 4 \cdot \left(\frac{s_k}{2}\right)^2 = 2 \cdot s_k \cdot s_i + s_k^2$$

$$\text{and} \quad \operatorname{area}(F_R) = 2 \cdot (H + W) \cdot \frac{s_k}{2} - 4 \cdot \left(\frac{s_k}{2}\right)^2 = (H + W) \cdot s_k - s_k^2.$$

By the inequality between arithmetic and quadratic mean, we know that

$$\frac{\sum_{i=1}^{n} s_i}{n} \leq \sqrt{\frac{\sum_{i=1}^{n} s_i^2}{n}} \leq \sqrt{\frac{1}{n}} \qquad \Rightarrow \qquad \sum_{i=1}^{n} s_i \leq \sqrt{n}.$$

We further have $H = \frac{F}{W}$ and $W \in [\frac{1}{10}, \sqrt{F}]$ since $\frac{1}{10} \leq W \leq H$, so by convexity of the function $[\frac{1}{10}, \sqrt{F}] \to \mathbb{R}^+, W \mapsto W + \frac{F}{W}$, which attains its minimum at $\sqrt{F}$ and is, therefore, monotonically decreasing, it follows that

$$W + H \leq \frac{1}{10} + \frac{F}{\frac{1}{10}} = 10F + \frac{1}{10} \leq \sqrt{n},$$

where the last inequality is implied by our choice of n. Moreover,

$$\sum_{i=n+1}^{k-1} \operatorname{area}(S'_i) = \sum_{i=n+1}^{k-1} (s_i + s_k)^2 \leq \sum_{i=n+1}^{k-1} (2s_i)^2 = 4 \sum_{i=n+1}^{k-1} s_i^2 \leq 4(c^2 - s_k^2),$$

because the sequence $(s_i)_{i \geq n+1}$ is non-increasing. Hence, we obtain

$$\begin{aligned} \operatorname{area}(M) &\geq F - 1 - \sum_{i=1}^{n} (2 \cdot s_i \cdot s_k + s_k^2) - ((W + H) \cdot s_k - s_k^2) - 4(c^2 - s_k^2) \\ &\geq F - 1 - 2\sqrt{n} \cdot s_k - n \cdot s_k^2 - \sqrt{n} \cdot s_k + s_k^2 - 4c^2 + 4s_k^2 \\ &> F - 1 - 4c^2 - 3\sqrt{n} \cdot s_k - n \cdot s_k^2 \end{aligned}$$

and we need to see that the latter term is non-negative whenever $s_k > 0$. To this end, we

first want to show that our choice of c implies $4 \cdot c^2 \leq F$. We have

$$\begin{aligned}
& 4 \cdot c^2 && \leq F && (1)\\
\Leftrightarrow\ & 4 \cdot \left(\sqrt{\left(\frac{3}{10}\right)^2 + \frac{F-1}{5}} - \frac{3}{10}\right)^2 && \leq F\\
\Leftrightarrow\ & 4 \cdot \left(\left(\frac{3}{10}\right)^2 + \frac{F-1}{5}\right) - 8 \cdot \sqrt{\left(\frac{3}{10}\right)^2 + \frac{F-1}{5}} \cdot \frac{3}{10} + 4 \cdot \left(\frac{3}{10}\right)^2 && \leq F\\
\Leftrightarrow\ & \frac{9}{25} + \frac{4}{5} \cdot F - \frac{4}{5} + \frac{9}{25} && \leq F + \frac{12}{5} \cdot \sqrt{\left(\frac{3}{10}\right)^2 + \frac{F-1}{5}}\\
\Leftrightarrow\ & \frac{18}{25} - \frac{20}{25} && \leq \frac{F}{5} + \frac{12}{5} \cdot \sqrt{\left(\frac{3}{10}\right)^2 + \frac{F-1}{5}}\\
\Leftrightarrow\ & -\frac{2}{25} && \leq \frac{F}{5} + \frac{12}{5} \cdot \sqrt{\left(\frac{3}{10}\right)^2 + \frac{F-1}{5}},
\end{aligned}$$

which is true since the left hand side is negative and the right hand side is positive as $F > 1$. Now, we show that $0 < s_k \leq \frac{c}{\sqrt{n}}$ and our assumptions on c and n imply that

$$F - 1 - 4c^2 - 3\sqrt{n} \cdot s_k - n \cdot s_k^2 \geq 0.$$

As $F - 1 - 4c^2 - 3\sqrt{n} \cdot x - n \cdot x^2 = 0$ if and only if

$$x = \frac{1}{\sqrt{n}} \cdot \left(-\frac{3}{2} - \sqrt{\frac{9}{4} + F - 1 - 4c^2}\right) \text{ or } x = \frac{1}{\sqrt{n}} \cdot \left(-\frac{3}{2} + \sqrt{\frac{9}{4} + F - 1 - 4c^2}\right),$$

where the discriminant is positive by (1), we know that $\text{area}(M) > 0$, provided that $s_k \leq \frac{1}{\sqrt{n}} \cdot \left(-\frac{3}{2} + \sqrt{\frac{9}{4} + F - 1 - 4c^2}\right)$. To this end, as $s_k \leq \frac{c}{\sqrt{n}}$, it suffices to see that we have $c \leq -\frac{3}{2} + \sqrt{\frac{9}{4} + F - 1 - 4c^2}$, or, equivalently, $c + \frac{3}{2} \leq \sqrt{\frac{9}{4} + F - 1 - 4c^2}$ respectively $(c + \frac{3}{2})^2 \leq \frac{9}{4} + F - 1 - 4c^2$ since $c > 0$. We get

$$\begin{aligned}
& \left(c + \frac{3}{2}\right)^2 && \leq \frac{9}{4} + F - 1 - 4c^2\\
\Leftrightarrow\ & c^2 + 3c + \frac{9}{4} && \leq \frac{9}{4} + F - 1 - 4c^2\\
\Leftrightarrow\ & 5c^2 + 3c - (F-1) && \leq 0\\
\Leftrightarrow\ & c^2 + \frac{3}{5}c - \frac{F-1}{5} && \leq 0,
\end{aligned}$$

which is equivalent to $-\frac{3}{10} - \sqrt{\left(\frac{3}{10}\right)^2 + \frac{F-1}{5}} \leq c \leq -\frac{3}{10} + \sqrt{\left(\frac{3}{10}\right)^2 + \frac{F-1}{5}}$ and follows from our choice of c and since the left hand term is negative.
Hence, whenever $s_k > 0$, the set of feasible midpoint locations for the respective square is non-empty, which concludes the proof. ◀

▶ **Lemma 4.** *Let c be defined as in Lemma 3 and let $\delta := \frac{F-1}{\frac{10F}{c^2} + \frac{1}{10}}$. Then any set of squares with total area V satisfying $c^2 \leq V \leq 1$ and maximum edge length at most δ can be packed into any rectangle of area $F \cdot V$ with greater edge length at most $10F$.*

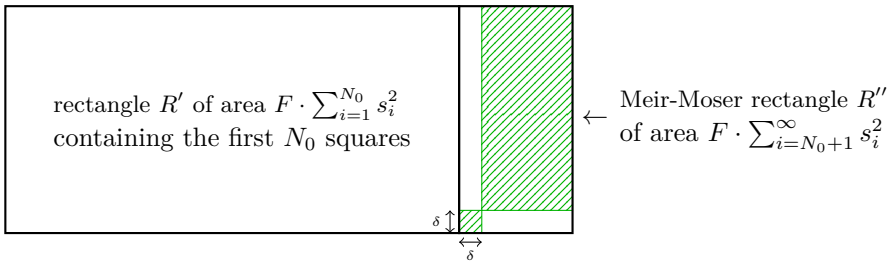


Figure 4 Gluing together R' and R'' provides the desired packing.

▶ **Lemma 5.** *Let c be as in Lemma 3, δ as in Lemma 4 and define $N_0 := \max\{1, \lfloor \frac{1}{\delta^2} \rfloor\}$. Assume that for any set S of at most N_0 squares of total area 1, there exists a rectangle R of area F into which the squares in S can be packed axis-parallel and internally disjoint. Then for any non-increasing sequence of non-negative numbers $(s_i)_{i\in\mathbb{N}^+}$ satisfying $s_1 \geq \frac{1}{10}$, $\sum_{i=1}^{\infty} s_i^2 = 1$ and $\sum_{i=N_0+1}^{\infty} s_i^2 \geq c^2$, there is a rectangle R of area F into which the squares with edge lengths given by $(s_i)_{i\in\mathbb{N}^+}$ permit an axis-parallel, internally disjoint packing.*

Proof. Let $(s_i)_{i\in\mathbb{N}^+}$ be as in the lemma. Note that since $s_1 \geq \frac{1}{10}$, the total area of the first N_0 squares is strictly positive. Apply the assumption stated for sets of at most N_0 squares to the set of squares the edge lengths of which arise from the sequence $(s_i)_{i=1}^{N_0}$ by normalizing the area $\sum_{i=1}^{N_0} s_i^2$ to 1, and then scale back. We obtain a rectangle R' of area $F \cdot \sum_{i=1}^{N_0} s_i^2 \leq F$ into which we can pack the squares with edge lengths $s_1, \ldots, s_{N_0}$. Let W' be the smaller edge length of R'. Then

$$\frac{1}{10} \leq s_1 \leq W' \leq \sqrt{F} \cdot \sqrt{\sum_{i=1}^{N_0} s_i^2} \leq \sqrt{F}.$$

Let $V := \sum_{i=N_0+1}^{\infty} s_i^2$ and let R'' be the rectangle with edge lengths W' and $\frac{FV}{W'}$. Then $c^2 \leq V \leq 1$ and for the greater edge length H'' of R'', we have

$$H'' = \max\left\{W', \frac{FV}{W'}\right\} \leq \max\left\{\sqrt{F}, \frac{F}{\frac{1}{10}}\right\} = 10F.$$

Additionally, $s_1 \geq \cdots \geq s_{N_0} \geq s_{N_0+1}$ and $\sum_{i=1}^{\infty} s_i^2 = 1$ yield $s_{N_0+1} \leq \frac{1}{\sqrt{N_0+1}} \leq \delta$, so by Lemma 4, we can pack the squares with edge lengths $s_{N_0+1}, s_{N_0+2}, \ldots$ into the rectangle R''. Gluing R' and R'' together at the edge of length W' yields a rectangle R of area

$$F \cdot \left(\sum_{i=1}^{N_0} s_i^2 + \sum_{i=N_0+1}^{\infty} s_i^2\right) = F$$

containing all squares (see Figure 4). ◀

▶ **Theorem 6.** *Let N_0 as in Lemma 5 and c as in Lemma 3 and define $N_1 := \lfloor \max\{N_0, \left(10F + \frac{1}{10}\right)^2, 100c^2\} \rfloor$ and $N := \lfloor \mathrm{e}^2 \cdot N_1 \rfloor$. Then any set of squares of total area 1 can be packed into some rectangle of total area F, provided this holds for any set of at most N squares of total area 1.*

Proof of Theorem 6. Note that for any natural number n, we have

$$\begin{aligned}\ln(n+1) &= \int_1^{n+1} \frac{1}{x}dx \leq \sum_{i=1}^{n}(i+1-i)\cdot\frac{1}{i} = \sum_{i=1}^{n}\frac{1}{i} = 1+\sum_{i=2}^{n}\frac{1}{i}\\ &= 1+\sum_{i=1}^{n-1}(i+1-i)\cdot\frac{1}{i+1} \leq 1+\int_1^n \frac{1}{x}dx = \ln(n)+1\\ \Rightarrow \ln(n+1) &\leq \sum_{i=1}^{n}\frac{1}{i} \leq \ln(n)+1.\end{aligned}$$

By our choice of N and since $N_1 \geq \lfloor N_0 \rfloor = N_0 \geq 1$, we have

$$\begin{aligned}\sum_{i=N_1+1}^{N}\frac{1}{i} &= \sum_{i=1}^{N}\frac{1}{i} - \sum_{i=1}^{N_1}\frac{1}{i} \geq \ln(N+1) - \ln(N_1) - 1 \geq \ln(\mathrm{e}^2 N_1) - \ln(N_1) - 1\\ &= \ln\left(\frac{\mathrm{e}^2 N_1}{N_1}\right) - 1 = \ln(\mathrm{e}^2) - 1 = 2 - 1 = 1.\end{aligned}$$

$$\text{In particular, } \sum_{i=N_1+1}^{N}\frac{c^2}{i} \geq c^2. \tag{2}$$

Consider any set of squares of total area 1 and the corresponding non-increasing sequence $(s_i)_{i\in\mathbb{N}^+}$ of edge lengths. If $s_1 \leq \frac{1}{10}$, we are done by Theorem 1.
So assume $s_1 > \frac{1}{10}$. If $\sum_{i=N_1+1}^{\infty} s_i^2 \geq c^2$, then we are done by Lemma 5 and our assumption since $N_0 \leq N_1$ because N_0 is integral and, therefore, also $\sum_{i=N_0+1}^{\infty} s_i^2 \geq c^2$. Hence, we can further assume that $\sum_{i=N_1+1}^{\infty} s_i^2 < c^2$. As $\sum_{i=N_1+1}^{N}\frac{c^2}{i} \geq c^2$ by (2), we cannot have $s_i \geq \frac{c}{\sqrt{i}}$ for all $N_1+1 \leq i \leq N$, so there is $N_1+1 \leq n \leq N$ with $s_n < \frac{c}{\sqrt{n}}$. By our packing assumption for sets of at most $N \geq n$ squares, we know that the squares with edge lengths $s_1, \ldots, s_{n-1}, \sqrt{1-\sum_{i=1}^{n-1} s_i^2} \geq s_n$ can be packed into some rectangle of area F, so this in particular also holds for the squares with edge lengths $s_1, \ldots, s_n$. But now, as we also have $\sum_{i=n+1}^{\infty} s_i^2 \leq \sum_{i=N_1+1}^{\infty} s_i^2 < c^2$ and $\frac{c}{\sqrt{n}} \geq s_n \geq s_{n+1} \geq s_{n+2} \geq \ldots$, we can apply Lemma 3, knowing that $n \geq N_1+1 \geq \max\{(10F+\frac{1}{10})^2, 100c^2\}$, to conclude the proof. ◀

For $F = \frac{2+\sqrt{3}}{3}$, a straightforward application of the previous arguments results in value of $N = 692,741,307$. With a little more effort, one can improve this to $N = 3,629,689$.

References

1. Stefan Hougardy. On packing squares into a rectangle. *Computational Geometry*, 44:456–463, 2011. doi:10.1016/j.comgeo.2011.05.001.
2. Aylin Ilhan. Das Packen von Quadraten in ein Rechteck. Diploma thesis, Universität Bonn, Forschungsinstitut für Diskrete Mathematik, March 2014.
3. Daniel Kleitman and Michael Krieger. Packing squares in rectangles I. *Annals of the New York Academy of Sciences*, 175:253–262, 1970. doi:10.1111/j.1749-6632.1970.tb56476.x.
4. Daniel Kleitman and Michael Krieger. An optimal bound for two dimensional bin packing. *16th Annual Symposium on Foundations of Computer Science*, pages 163–168, 1975. doi:10.1109/SFCS.1975.6.
5. Aram Meir and Leo Moser. On packing of squares and cubes. *Journal of Combinatorial Theory*, 5(2):126–134, 1968. doi:10.1016/S0021-9800(68)80047-X.

6 John Moon and Leo Moser. Some packing and covering theorems. In *Colloquium mathematicum*, volume 17, pages 103–110. Institute of Mathematics Polish Academy of Sciences, 1967.

7 Leo Moser. Poorly formulated unsolved problems of combinatorial geometry. *Mimeographed list*, 1966.

8 Pavel Novotný. A note on a packing of squares. *Stud. Univ. Transp. Commun. Zilina Math.-Phys. Ser.*, 10:35–39, 1995.

9 Pavel Novotný. On packing of squares into a rectangle. *Archivum Mathematicum (BRNO)*, 32:75–83, 1996.

10 Pavel Novotný. On packing of four and five squares into a rectangle. *Note di matematica*, 19/2:199–206, 1999. doi:10.1285/i15900932v19n2p199.

11 Alexander Platz. A proof of Moser's square packing problem for small instances. Master's thesis, Universität Bonn, Forschungsinstitut für Diskrete Mathematik, August 2016.

12 Jan Zernisch. A generalization of a theorem of Kleitman and Krieger. *International Journal of Computational Geometry & Applications*, 22(02):167–185, 2012. doi:10.1142/S0218195912500033.

On 4-Crossing-Families in Point Sets and an Asymptotic Upper Bound*

Oswin Aichholzer[1], Jan Kynčl[2], Manfred Scheucher[1,3], and Birgit Vogtenhuber[1]

1 **Institute of Software Technology, Graz University of Technology, Austria**
oaich@ist.tugraz.at, bvogt@ist.tugraz.at
2 **Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic**
kyncl@kam.mff.cuni.cz
3 **Institut für Mathematik, Technische Universität Berlin, Germany**
scheucher@math.tu-berlin.de

Abstract

A k-crossing family in a point set S in general position is a set of k segments spanned by points of S such that all k segments mutually cross. In this short note we present two statements on crossing families which are based on sets of small cardinality: (1) Any set of at least 15 points contains a crossing family of size 4. (2) There are sets of n points which do not contain a crossing family of size larger than $4\lceil \frac{n}{20} \rceil < \frac{n}{5} + 4$. Both results improve the previously best known bounds.

1 Introduction

Let S be a set of n points in the Euclidean plane *in general position*, that is, no three points in S are collinear. A *segment* of S is a line segment with its two endpoints (which we will also call vertices) being points of S.

▶ **Definition 1.1.** A *k-crossing family* in a point set S is a set of k segments spanned by points of S such that all k segments mutually cross in their interior.

For a point set S, let $\mathrm{cf}(S)$ be the maximum size of a crossing family in S, and let $\mathrm{cf}(n)$ be the minimum of $\mathrm{cf}(S)$ over all point sets S of cardinality n in general position.

It is easy to see that $\mathrm{cf}(n)$ is a monotone function. From the fact that the complete graph with 5 vertices is not planar it follows that any set of $n \geq 5$ points has a crossing family of size at least 2. In [1] it is shown that every set with $n \geq 10$ points admits a crossing family of size 3. The result is based on analyzing the set of all order types of size 10. The bound on n is tight, that is, there exist 12 order types of 9 points which do have a maximal crossing family of size 2. One such set is shown in Figure 1.

Aronov et al. [3] proved in 1994 the existence of crossing families of size $\sqrt{n/12}$ for every set of n points, which until recently was the best general lower bound. Their proof relies on mutually avoiding sets, for which the given bound is asymptotically tight.

* Research for this article was initiated in the course of the bilateral research project "Erdős–Szekeres type questions for point sets" between Graz and Prague, supported by the OEAD project CZ 18/2015 and project no. 7AMB15A T023 of the Ministry of Education of the Czech Republic. J.K. was also supported by the grant no. 21-32817S of the Czech Science Foundation (GAČR) and by Charles University project UNCE/SCI/004. B.V. partially supported by Austrian Science Fund within the collaborative DACH project *Arrangements and Drawings* as FWF project I 3340-N35.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

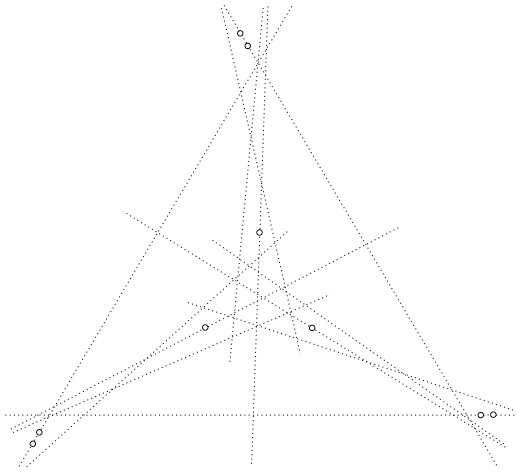


Figure 1 A set S of 9 points which does not contain a 3-crossing family, that is, $\mathrm{cf}(S) = 2$.

Only in 2019 Pach, Rubin, and Tardos [9] showed in a breakthrough result that any set of n points in general position in the plane contains a crossing family of size $n^{1-o(1)}$. This almost shows the generally accepted conjecture that $\mathrm{cf}(n)$ should be $\Theta(n)$.

The conjecture is also supported by the currently best upper bounds. Recently, Evans and Saeedi [6] showed that $\mathrm{cf}(n) \leq 5\lceil \frac{n}{24} \rceil$. We will improve that bound to $\mathrm{cf}(n) \leq 4\lceil \frac{n}{20} \rceil$ in Section 3. In the next section we start by showing that $\mathrm{cf}(15) = 4$.

2 Sets of 15 points always contain a 4-crossing family

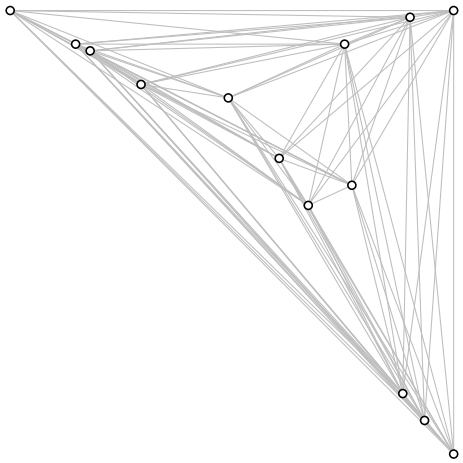
From the mentioned result $\mathrm{cf}(10) = 3$ [1] we know that also any set of 11 points contains a 3-crossing family, and no such set can contain a crossing family of size more than 5 (as at least $2k$ points are needed for a k-crossing family). The following table shows how the maximal size of a crossing family is distributed among all combinatorially different point sets of size 11 and was computed with the help of the database of all order types of size 11 obtained in [2].

k	number of order types
3	63 978 178
4	1 783 117 647
5	487 417 082
total	2 334 512 907

Table 1 Number of realizable order types of size 11 containing at most a k-crossing family.

To obtain the largest point set containing no crossing family of size 4, we made a complete abstract order type extension from $n = 11$ to $n = 15$. The database of all realizable order types of cardinality 11 contains 2 334 512 907 sets [2], of which 63 978 178 (about 2.7 %) contain no 4-crossing family; see Table 1. Since adding points to an existing set can never decrease the size of the maximal crossing family, we need to consider only those sets.

The approach of extending order types in an abstract way as described in [2] has the advantage that there is no need to realize the obtained sets, as we actually are interested in



coordinates:

61	66
61	0
57	5
54	9
47	40
41	37
37	44
30	53
46	61
18	55
55	65
11	60
9	61
0	66

Figure 2 A set S of 14 points which does not contain a 4-crossing family, that is, $\mathrm{cf}(S) = 3$.

the smallest cardinality where no such sets exist. This avoids dealing with the notoriously hard problem of realizing abstract order types, which is know to be $\exists\mathbb{R}$-hard [8]. The extension is done iteratively by adding one more (abstract) point in each step. Afterwards each obtained abstract order type is checked for the maximum size of a crossing family, and if this is larger than 4 the abstract order type is discarded.

After the first three rounds we obtained 2 727 858 abstract order types of cardinality 14 which do not contain a 4-crossing family. All these sets were extended by one further point (in the abstract setting), but all resulting sets contained a 4-crossing family. Thus, the largest possible set with no 4-crossing family has size at most 14. The whole process of abstract extension took about 100 hours, computed in parallel on 40 standard CPUs.

To show that the obtained bound is best possible it is sufficient to realize at least one generated abstract order type of size 14, and such an example is given in Figure 2. Together with the set of 20 points with no 5-crossing family depicted in Figure 5, we conclude the following.

▶ **Theorem 2.1.** *Every set of at least* 15 *points in the plane in general position contains a* 4*-crossing family. Moreover, we have*

$$\mathrm{cf}(n) = 3 \text{ for } 10 \leq n \leq 14 \quad \text{and} \quad \mathrm{cf}(n) = 4 \text{ for } 15 \leq n \leq 20.$$

Besides the above described computer proof, we have also developed a SAT framework which allowed us to verify $\mathrm{cf}(15) > 3$ within less than 2 CPU days using the SAT solver CaDiCaL [5]. We have also used this framework to verify $\mathrm{cf}(10) > 2$ [1]. The python program creating the instance is available on our supplemental website [11].

The idea behind the SAT model is very similar as in [12]: We assume towards a contradiction that $\mathrm{cf}(15) \leq 3$, that is, there is a set of 15 points with no 4-crossing family. We have Boolean variables X_{abc} to indicate whether three points a, b, c are positively or negatively oriented. As outlined in [12], these variable have to fulfill the signotope axioms [7, 4]. Based on the variables for triple orientations, we then assign auxiliary variables $Y_{ab,cd}$ to indicate whether the two segments ab, cd cross. Finally we assert that for any set of four segments with pairwise distinct endpoints, at least one pair of them does not cross. As the SAT solver CaDiCaL terminates with "unsatisfiable", no such point set exists, and hence $\mathrm{cf}(15) > 3$.

3 Small sets and an upper bound

The following theorem was already implicitly used by Aronov et al. [3, Section 6] in their discussion, where they stated the upper bound $\mathrm{cf}(n) \leq \frac{n}{4}$. It can also be found as Lemma 3 in [6]. Since in [3] no proof is given and the proof from [6] appears to be incomplete (see Figure 3), we include a full proof here.

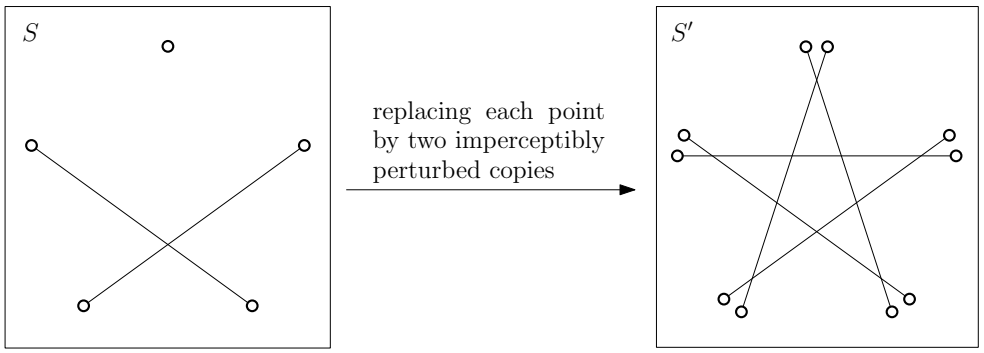


Figure 3 Five points in convex position have at most a 2-crossing family (left). Replacing each point by two imperceptibly perturbed copies with no prescribed structure as in [6] may result in a set of 10 points with a crossing family of size larger than 4 (right). The process needs to avoid generating odd cycles of pairwise crossing edges; see the proof of Theorem 3.1 for details.

▶ **Theorem 3.1.** *Let $S \subset \mathbb{R}^2$ be a set of n points in general position with $\mathrm{cf}(S) = k$. Then for any $N \geq n$ there exists a set $S' \subset \mathbb{R}^2$ of N points in general position with $\mathrm{cf}(S') \leq k\lceil \frac{N}{n} \rceil$.*

For our proof of Theorem 3.1, we will use a simple property of geometric thrackles. A *geometric thrackle* is a geometric graph such that each pair of edges (drawn as line segments) either meets at a common vertex or crosses properly. Woodall proved that a graph can be drawn as a geometric thrackle if and only if it is a a subgraph of a graph obtained by attaching leaves (vertices of degree 1) to the vertices of an odd cycle [13, Theorem 2]. We will need only the following weaker characterization.

▶ **Lemma 3.2.** *A geometric thrackle T contains no even cycles.*

A strenghtening to monotone thrackles was proved by Pach and Sterling [10]. Since the proof for geometric thrackles is substantially simpler, we include it here to keep this note self-contained.

Proof of Lemma 3.2. Assume there exists an even cycle $C = p_0, p_1, \ldots, p_n$ for some even $n \geq 4$ with $p_0 = p_n$ and $\overline{p_{i-1}p_i} \in T$ for $i = 1, \ldots, n$. We set $p_i = p_j$ for $i = j \pmod n$. Consider a line segment $\ell = \overline{p_i p_{i+1}}$ in C. The supporting line through ℓ divides the plane into two half-planes. Since T is a thrackle, the previous segment $\overline{p_{i-1}p_i}$ and the next segment $\overline{p_{i+1}p_{i+2}}$ cross, and thus p_{i-1} and p_{i+2} lie in the same half-plane. Moreover, since all segments in the path $P = p_{i+2}, p_{i+3}, \ldots, p_{i-1}$ cross the segment $\overline{p_i p_{i+1}}$, P is an alternating path with respect to the side of the half-plane, and hence P has even length $|P|$, that is, an even number of edges. Since the cycle C has length $|C| = |P| + 3$ this is a contradiction, because C was assumed to have even length. ◀

Proof of Theorem 3.1. Let S be a set of n points in general position in the plane and let $m = \lceil \frac{N}{n} \rceil$. Without loss of generality we may assume that $N = mn$; in case $N < mn$ we

may later remove some points from the constructed point set. We can also assume that all points of S have distinct x- and y-coordinates; otherwise we slightly rotate S. Our aim is to construct a set S' of mn points by creating m *copies* of each point from S, such that the following two properties hold:

(S1) a line segment between two copies of $p \in S$ only intersects line segments incident to another copy of p,
(S2) all copies of a point are almost on a horizontal line; that is, if $p \in S$ is above (below) $q \in S$ then any line through two different copies of p is above (below) any copy of q.

To this end, we place the m copies of a point $p = (x, y)$ at $p_i = (x + i\varepsilon, y + (i\varepsilon)^2)$ for $i = 0, \dots, m-1$. For sufficiently small $\varepsilon > 0$, all points from S' have distinct x- and y-coordinates and the above conditions are fulfilled.

Let F' be a maximum crossing family in S'. We will show how to find a crossing family F in S of size at least $|F'|/m$. If $|F'| \leq m$, we are clearly done since any segment in S is a crossing family of size 1. Hence assume that F' contains more than m segments. Then no segment $f' \in F'$ can be incident to two copies of the same point of S due to property (S1). Thus every segment $f' \in F'$ connects (copies of) two distinct points of S.

Let F be the set of segments (without multiplicity) on S induced by F' and by contracting the m copies of p_i to p, for each $p \in S$. Formally, $F = \{\overline{pq} \mid \exists i, j \colon \overline{p_i q_j} \in F'\}$; Figure 4 gives an illustration. Let G_F be the geometric graph induced by F, which has the set of end points of F as (drawn) vertices and F as (drawn) edges. More formally, $G_F = (V, E, \phi, \psi)$ with $V = \{p \in S \mid \exists q \colon \overline{pq} \in F\}$, $E = \{\{p, q\} \mid \overline{pq} \in F\}$, $\phi : V \to \mathbb{R}^2, p \mapsto p$, and $\psi \colon \{p, q\} \mapsto \overline{pq}$ for any $\{p, q\} \in E$. By construction G_F is a geometric thrackle, and due to Lemma 3.2, G_F contains no even cycles. Observe that according to property (S2), the neighbors of a vertex p in G_F can either be all above p or all below p, as no segment of F connected from above to a copy p_i can cross a segment of F connected from below to a copy p_j. As a consequence, G_F is bipartite and thus contains no odd cycles. Hence, G_F is acyclic; equivalently, a forest.

As long as G_F contains vertices of degree larger than 1, we continue as follows: Let $p \in V(G_F)$ be a leaf incident to a vertex $q \in V(G_F)$ of degree larger than 1. We construct F'' by removing all segments in F' incident to copies of q, and by inserting segments connecting all copies of q to copies of p in the way such that all those segments cross; Figure 4 gives an illustration of this modification. By construction, $|F''| \geq |F'|$ holds and thus F'' is another maximal crossing family in S'. We can replace F' by F''.

We can iteratively repeat this process, and in every step the number of vertices of degree larger than 1 strictly decreases. Therefore we can assume that G_F contains no vertices of degree greater than 1. As a consequence, F is a (not necessarily maximal) crossing family in S with $|F| \geq |F'|/m$. Therefore, $\mathrm{cf}(S') \leq |F'| \leq m \cdot |F| \leq m \cdot \mathrm{cf}(S) = mk$. ◀

From the point set depicted in Figure 1, it follows by Theorem 3.1 that there are sets of n points with no crossing family larger than $2\lceil \frac{n}{9} \rceil$. Evans and Saeedi [6] constructed a set of 24 points with no crossing family of size 6 or more, which yields the upper bound $\mathrm{cf}(n) \leq 5\lceil \frac{n}{24} \rceil$ presented there.

k	1	2	3	4	5	6	7	8	9	10
Currently best example	4	9	14	20	25	29	34	40	44	50

Table 2 Largest known point sets S_k containing at most a k-crossing family, that is, with $\mathrm{cf}(S_k) = k$. For $k \leq 3$ the sets are best possible.

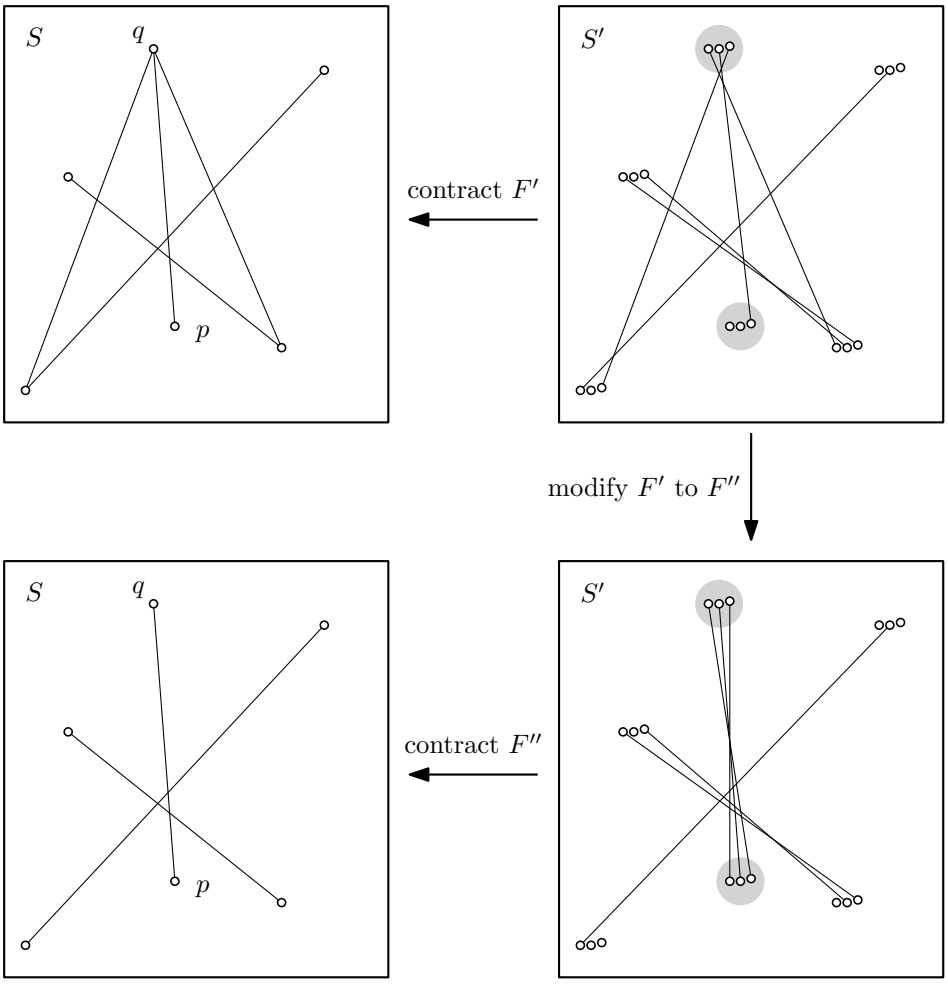


Figure 4 An illustration of the contracting process to attain F from F', and an illustration of the modification of F' in which all copies of p and q are "connected" to each other.

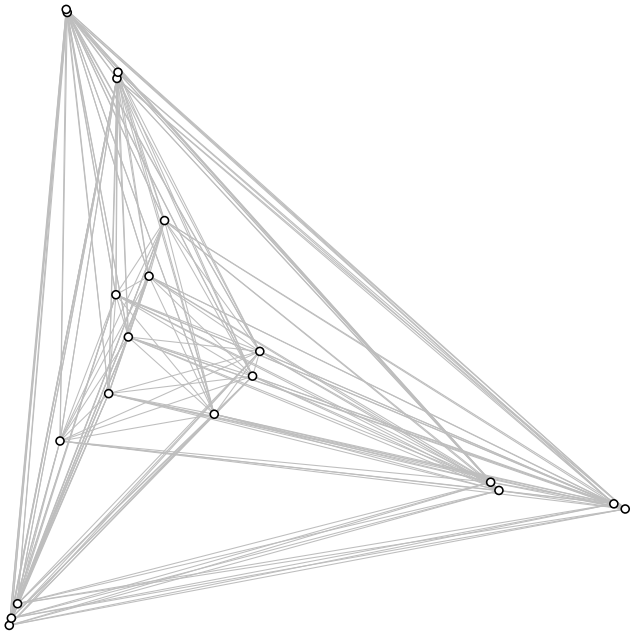
To further improve this bound we searched for sets with small crossing families. For $k \geq 5$ we partially extended several smaller sets (for example by doubling the number of points similarly to the process described in the proof of Theorem 3.1) and used heuristics such as simulated annealing to optimize them. Table 2 summarizes the sizes of the currently largest sets containing at most a k-crossing family for k up to 10.

Using Theorem 3.1 together with the sets of 20 points containing no 5-crossing family (see Figure 5) we get the bound $\mathrm{cf}(n) \leq 4\lceil \frac{n}{20} \rceil$. Also we have a set of 25 points containing no 6-crossing family (see Figure 6), which implies $\mathrm{cf}(n) \leq 5\lceil \frac{n}{25} \rceil$ and therefore gives a slightly better upper bound for certain values of n.

▶ **Corollary 3.3.** *We have* $\mathrm{cf}(n) \leq \min\{4\lceil \frac{n}{20} \rceil, 5\lceil \frac{n}{25} \rceil\} < \frac{n}{5} + 4$.

4 Conclusion

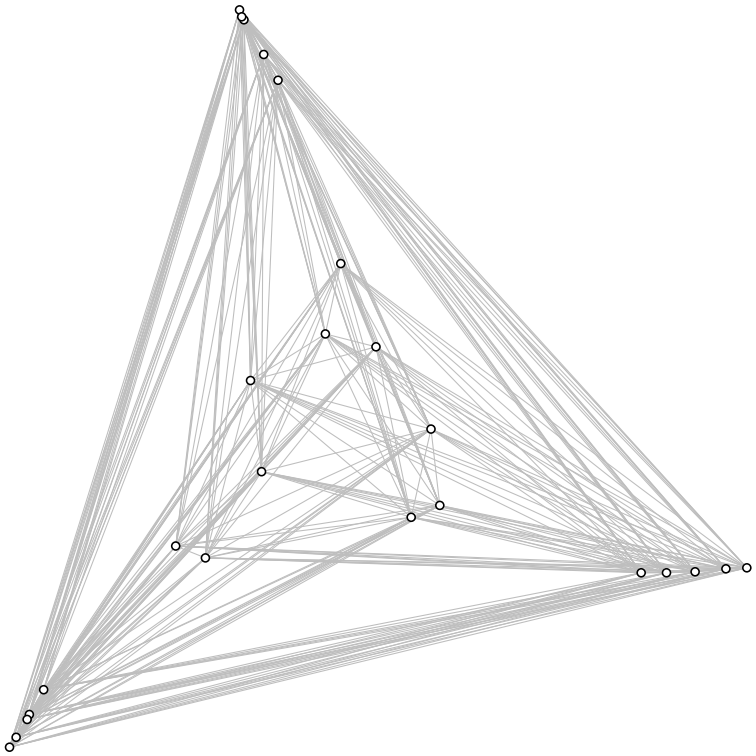
Based on exhaustive abstract extension of order types we showed that $\mathrm{cf}(15) = 4$. We conjecture that every set of 21 or more points contains a crossing family of size 5. We plan



coordinates:

595	113
0	0
2	7
8	21
49	179
473	131
465	139
96	225
198	205
115	280
235	242
103	321
242	266
584	118
135	339
150	393
104	531
105	537
56	595
55	598

Figure 5 A set S of 20 points with no 5-crossing family.



coordinates:

209	744
670	181
651	180
623	177
597	176
574	176
231	699
244	673
213	734
383	321
333	404
391	244
301	488
365	232
211	737
287	417
229	278
219	370
178	191
151	203
31	58
18	33
16	28
6	10
0	0

Figure 6 A set S of 25 points with no 6-crossing family.

to use our SAT framework to either settle the conjecture or improve the upper bound on the size of a crossing family to $4\lceil \frac{n}{21} \rceil$.

Acknowledgements. This work was initiated in April 2015 during the Graz–Prague bilateral project *Erdős-Szekeres type questions for point sets*. We thank all the members of this project for the good atmosphere as well as discussions on the topic.

References

1 Oswin Aichholzer and Hannes Krasser. The Point Set Order Type Data Base: A Collection of Applications and Results. In *Proc.* 13^{th} *Annual Canadian Conference on Computational Geometry CCCG 2001*, pages 17–20, 2001.

2 Oswin Aichholzer and Hannes Krasser. Abstract order type extension and new results on the rectilinear crossing number. *Comput. Geom.*, 36(1):2–15, 2007. doi:10.1016/j.comgeo.2005.07.005.

3 Boris Aronov, Paul Erdős, Wayne Goddard, Daniel J. Kleitman, Michael Klugerman, János Pach, and Leonard J. Schulman. Crossing families. *Combinatorica*, 14(2):127–134, 1994. doi:10.1007/BF01215345.

4 Martin Balko, Radoslav Fulek, and Jan Kynčl. Crossing numbers and combinatorial characterization of monotone drawings of K_n. *Discrete Comput. Geom.*, 53(1):107–143, 2015. doi:10.1007/s00454-014-9644-z.

5 Armin Biere. CaDiCaL at the SAT Race 2019. In *Proc. of SAT Race 2019 – Solver and Benchmark Descriptions*, volume B-2019-1 of *Department of Computer Science Series*, pages 8–9. University of Helsinki, 2019. https://researchportal.helsinki.fi/en/publications/proceedings-of-sat-race-2019-solver-and-benchmark-descriptions.

6 William Evans and Noushin Saeedi. On problems related to crossing families. arXiv:1906.00191v1, 2019.

7 Stefan Felsner and Helmut Weil. Sweeps, arrangements and signotopes. *Discrete Appl. Math.*, 109(1-2):67–94, 2001. 14th European Workshop on Computational Geometry CG'98 (Barcelona). doi:10.1016/S0166-218X(00)00232-8.

8 Nikolai Mnëv. On manifolds of combinatorial types of projective configurations and convex polyhedra. In *Soviet Math. Doklady*, volume 32, pages 335–337, 1985.

9 János Pach, Natan Rubin, and Gábor Tardos. Planar point sets determine many pairwise crossing segments. In *STOC'19—Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 1158–1166. ACM, New York, 2019. doi:10.1145/3313276.3316328.

10 János Pach and Ethan Sterling. Conway's conjecture for monotone thrackles. *Amer. Math. Monthly*, 118(6):544–548, 2011. doi:10.4169/amer.math.monthly.118.06.544.

11 Manfred Scheucher. Webpage: On 4-Crossing-Families in Point Sets and an Asymptotic Upper Bound. http://page.math.tu-berlin.de/~scheuch/suppl/crossing_families.

12 Manfred Scheucher. Two disjoint 5-holes in point sets. *Comput. Geom.*, 91:101670, 12, 2020. doi:10.1016/j.comgeo.2020.101670.

13 D. R. Woodall. Thrackles and deadlock. In *Combinatorial Mathematics and its Applications (Proc. Conf., Oxford, 1969)*, pages 335–347. Academic Press, London, 1971.

Collapses of higher order Delaunay complexes*

Mickaël Buchet[1], Bianca B. Dornelas[2], and Michael Kerber[3]

1 **Institute of Geometry, TU Graz, Austria**
buchet@tugraz.at
2 **Institute of Geometry, TU Graz, Austria**
bdornelas@tugraz.at
3 **Institute of Geometry, TU Graz, Austria**
kerber@tugraz.at

Abstract

Bauer and Edelsbrunner [2] showed that the Delaunay-Čech complex collapses to the Delaunay complex for point sets in general position. We extend this result by showing that the kth order Delaunay-Čech complex collapses to the kth order Delaunay complex in the plane. We do this by building a sequence of simplicial collapses from the former to the latter.

1 Introduction

The Delaunay triangulation of a finite set of sites is among the most famous concepts in computational geometry [1, 6]. It can be used to define *Alpha complexes* by the removal of "large" simplices with respect to a threshold r. Considering all these thresholds leads to the concept of the *Delaunay filtration* (also known as the *alpha filtration*), a sequence of nested simplicial complexes that represents the input on multiple scales. A slight variation of the Delaunay filtrations are *Delaunay-Čech filtrations*, where the size of a simplex is not determined by its circumradius, but the radius of the minimum enclosing ball of the sites. In general, this results in a larger complex for every threshold r.

These concepts can be extended to the higher order case, where vertices are defined by k-subsets of sites instead of a single site. *Higher order Delaunay complexes* have simplices defined with spheres containing at most $(k-1)$ sites in their interior. The restriction to simplices with sphere radius smaller or equal to a threshold r yields an analogue of Delaunay filtrations for the kth order Delaunay complex. Similarly, Delaunay-Čech filtrations are extended to order k by considering the minimum enclosing ball of all k-subsets involved in a simplex.

Contribution. We show that, for point sets in general position in $\mathbb{R}^2$, the kth order Delaunay-Čech complex with radius r collapses to the kth order Delaunay complex with radius r through a sequence of elementary simplicial collapses [4, Sec.3.4]. To prove that, we partition the "excess" simplices in the Delaunay-Čech complex into intervals, and collapse these intervals in sequence.

As a consequence, the kth order Delaunay-Čech and Delaunay complexes are homotopically equivalent for every r. Moreover, the homotopy equivalence is realized by the inclusion map, immediately implying that the kth order Delaunay-Čech and Delaunay filtrations are weakly equivalent in the model-categoric sense (compare [3]) and, in particular, give rise to the same persistence diagram. The filtration values for the kth order Delaunay-Čech

* Supported by the Austrian Science Fund (FWF): W1230.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

complex are determined by minimal enclosing balls, which is simpler than considering both inclusion and exclusion constraints as is needed for the kth order Delaunay complex. We speculate that this simplicity may lead to computational advantages, even though state-of-the-art methods to compute Delaunay filtrations use a slight variation of the mini-ball algorithm [5].

Related Results. Our result partially generalizes the result by Bauer and Edelsbrunner [2]. They showed that, in any dimension, the 1st order Delaunay-Čech complex collapse to the 1st order Delaunay complex.

2 Background

Let $\mathcal{V}$ be a finite collection of elements called *vertices*. An *(abstract) simplex* σ of dimension n is a subset of $\mathcal{V}$ with $n+1$ elements. If $\sigma = \{V_0, V_1, \cdots, V_n\}$, we write $\sigma = V_0V_1 \cdots V_n$ and say that each V_i is a vertex of σ. A *face* of σ is a simplex spanned by a subset of the vertices of σ and a coface is a simplex spanned by a set containing all the vertices of σ. An *(abstract) simplicial complex* is a finite collection of simplices, K, such that for every simplex $\sigma \in K$, every face of σ is in K.

Free Collapses. Given $\sigma, \tau \in K$, if σ is a coface of τ with $\dim \sigma = \dim \tau + 1$, τ is a *facet* of σ. If in addition σ is the only coface of τ in K, τ is *free*. An *elementary collapse* in K is the removal of a pair (τ, σ), where σ has maximal dimension inside K and τ is a free facet of σ. We write $K \searrow K \setminus \{\tau, \sigma\}$. A *collapse* is a sequence of elementary collapses $K_0 \searrow K_1 \searrow \cdots \searrow K_m$ such that $K_{i+1} = K_i \setminus \{\tau_i, \sigma_i\}$. We write $K_0 \searrow K_m$. The collapses define a deformation retraction from K_0 to K_m hence both complexes are homotopically equivalent [4].

An *interval* in the face relation of K is a subset of the form $[P, R] = \{Q \mid P \subseteq Q \subseteq R\}$. Intervals with only one simplex are called *singular*, its simplex being a *critical simplex*.

The kth Order Delaunay Complex. We assume $X \subseteq \mathbb{R}^2$ to be a finite set in *general position*: no 3 points of X lie on a common line and no 4 points lie on a common circle. Elements of X are called *sites* and $A \subseteq X$ with $|A| = k$ is called a *k-subset*.

Given a circle S in $\mathbb{R}^2$, denote the set of *sites on* S as $\operatorname{On} S$ and the set of *sites inside* (but not on) S as $\operatorname{In} S$. $\operatorname{Incl} S = \operatorname{In} S \cup \operatorname{On} S$ is the collection of *sites included* by S. For a point $x \in \mathbb{R}^2$, let $S(x)$ denote the smallest circle centered at x that includes at least k sites. We say that x is *near* to a k-subset A of X if $A \subseteq \operatorname{Incl} S(x)$ and no other site is inside $S(x)$, i.e. there is no $p \in X \setminus A$ with $p \in \operatorname{In} S(x)$.

▶ **Definition 2.1.** Let $\mathcal{V}$ be the collection of all k-subsets with at least one near point. We say that $\sigma \subseteq \mathcal{V}, \sigma = \{A_0, A_1, \cdots, A_m\}$, is a *$k$-Delaunay simplex* of dimension m if there exists $x \in \mathbb{R}^2$ such that x is near to A_i, for all $0 \leq i \leq m$. The *kth order Delaunay complex* of X, $\operatorname{Del}^k(X)$, is the collection of all k-Delaunay simplices of X.

See Figure 1 for an example. The kth order Delaunay complex of X can also be interpreted as the nerve of the kth order Voronoi diagram of X, i.e. the diagram obtained by partitioning the space into regions with same k-closest sites.

The smallest circle $S(x)$, over all choices of points near to all of σ's vertices, is the *k-Delaunay circle* of σ. We often omit the mention of k. The *Higher Order Delaunay radius function* is the function s_{Del} associating to each simplex the radius of its Delaunay circle.

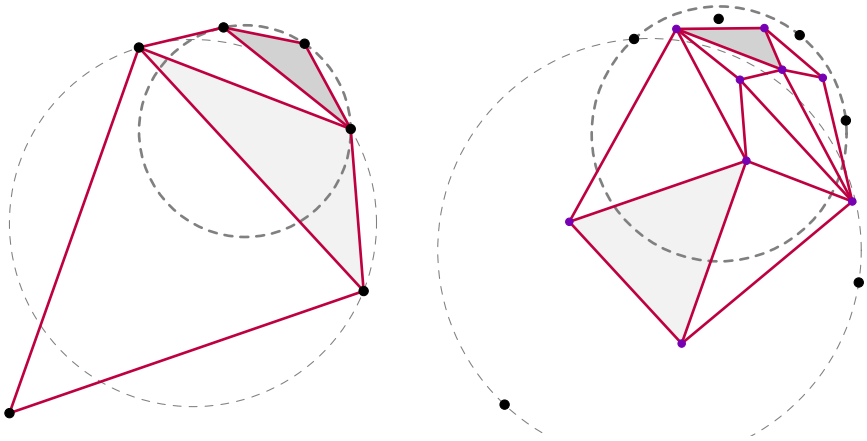


Figure 1 Examples of order 1 (left) and order 2 (right) Delaunay complexes. Dashed circles are the k-Delaunay circles of the filled triangles.

▶ **Definition 2.2.** The *kth order Delaunay complex with radius r*, denoted $\mathrm{Del}_r^k(X)$, is the collection of all k-Delaunay simplices whose k-Delaunay circle has radius smaller or equal to $r \in \mathbb{R}^+$, that is $\mathrm{Del}_r^k(X) = \{\sigma \in \mathrm{Del}^k(X) \mid s_{Del}(\sigma) \leq r\}$.

The kth Order Delaunay-Čech Complex. Given a simplex $\sigma \in \mathrm{Del}^k(X)$, we define its *Čech radius* as the radius of the minimal enclosing ball of all the involved sites, called the *k-Čech circle*. The Čech radius function is the function $s_{Čech}$ associating to each $\sigma \in \mathrm{Del}^k(X)$ its Čech radius. A Čech circle always exists and is the smallest circumcircle of its boundary sites (see for example [4]).

▶ **Definition 2.3.** Let $r \in \mathbb{R}^+$. The *kth order Delaunay-Čech complex with radius r*, denoted $\mathrm{DelČech}_r^k(X)$, is the set of all simplices of $\mathrm{Del}^k(X)$ with Čech radius at most r, that is

$$\mathrm{DelČech}_r^k(X) = \{\sigma \in \mathrm{Del}^k(X) \mid s_{Čech}(\sigma) \leq r\}.$$

$\mathrm{Del}_r^k(X)$ is a subset of $\mathrm{DelČech}_r^k(X)$ because $s_{Čech}(\sigma) \leq s_{Del}(\sigma)$ for any $\sigma \in \mathrm{Del}^k(X)$. Moreover, general position implies that there are no two different Čech or Delaunay circles with same radii and that the highest dimensional simplices are triangles.

Front and Back. Given a subset of 2 or 3 sites, $P \subseteq X$, there is a unique smallest circumcircle S for P. This circle S has the property that its center z can be expressed as an affine combination of the points in P:

$$z = \sum_{x \in \mathrm{On}\, S} \lambda_x x \text{ with } 1 = \sum_{x \in \mathrm{On}\, S} \lambda_x.$$

We define the *front face* and the *back face* of $\mathrm{On}\, S$ as

$$\mathrm{Front}\, S = \{x \in \mathrm{On}\, S \mid \lambda_x > 0\} \quad \text{and} \quad \mathrm{Back}\, S = \{x \in \mathrm{On}\, S \mid \lambda_x < 0\}.$$

General position implies that the coefficients are non-zero and $|\mathrm{Back}\, S| \leq 1$ in the plane. Front sites can be interpreted as those which are visible from the center, while the back site is hidden by some face of the convex hull $\mathrm{conv}(\mathrm{On}\, S)$, as depicted in Figure 2 (left).

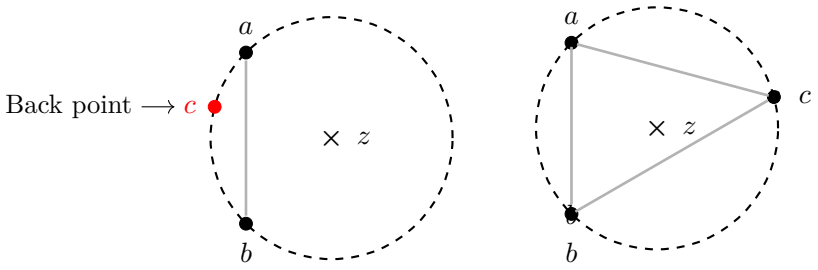


Figure 2 Left: circle with non-empty back set. Right: circle with empty back set.

3 Collapsing Delaunay-Čech to Delaunay

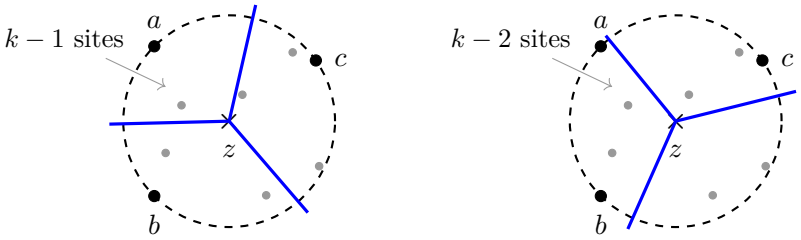
We want to collapse the simplices in $\text{DelČech}_r^k(X)$ that do not exist in $\text{Del}^k(X)$, which are precisely those simplices for which Delaunay and Čech radii differ. We define that a simplex $\sigma \in \text{Del}^k(X)$ is a *bad simplex* if $s_{Del}(\sigma) \neq s_{\check{C}ech}(\sigma)$. It is called *good* otherwise. The next lemma gives a geometric characterization of bad simplices.

▶ **Lemma 3.1.** *Let S be the k-Delaunay circle of $\sigma \in \text{Del}^k(X)$. Then σ is a bad simplex if and only if* $\text{Back}\, S \neq \emptyset$.

Proof. $\Rightarrow$| If $\text{Back}\, S = \emptyset$, the center of S is in the convex hull of the front points. In this case, S is also the minimal enclosing ball of the point set, hence the k-Čech circle.

$\Leftarrow$| If $\text{Back}\, S \neq \emptyset$, the center z of S is not inside the convex hull of its boundary points and there is an hyperplane $\mathcal{H}$ with z to one side and $\text{On}\, S$ to the other side. If we slightly move z in the direction of $\mathcal{H}$, reducing the radius to preserve the intersection with $\mathcal{H}$, then this is a smaller enclosing ball of $\text{On}\, S \cup \text{In}\, S$. Hence the k-Čech circle is smaller than S. ◀

Let $\sigma \in \text{Del}^k(X)$ be a triangle. Then, there is a unique point that is near to all three k-subsets defining σ, the Voronoi vertex z at the intersection of the three dual k-Voronoi regions. The smallest circle S centered at z that includes all involved sites in σ is its k-Delaunay circle. Call the boundary sites a, b, c and $P = \text{In}\, S$. There are two cases: $|P| = k-1$, in which case the vertices of σ are $P \cup \{a\}, P \cup \{b\}, P \cup \{c\}$ (Figure 3a), or $|P| = k-2$, in which case the vertices of σ are $P \cup \{a, b\}, P \cup \{a, c\}, P \cup \{b, c\}$ (Figure 3b). We call σ a $(k-1)$-triangle or a $(k-2)$-triangle, respectively.



(a) $(k-1)$-triangle **(b)** $(k-2)$-triangle

Figure 3 Possible k-Delaunay triangles and their k-Voronoi regions, delimited by the edges.

Consider σ a Delaunay simplex. Another simplex $\omega \in \text{Del}^k(X)$ is said to be *linked* to σ if ω is a face or a coface of σ, $s_{Del}(\sigma) = s_{Del}(\omega)$ and $s_{\check{C}ech}(\sigma) = s_{\check{C}ech}(\omega)$. We use this concept to construct the desired intervals in $\text{Del}^k(X)$.

▶ **Lemma 3.2.** *Let $\sigma \in \mathrm{Del}^k(X)$ be a bad 2-simplex and S be its Delaunay circle, with* $\mathrm{Front}\, S = \{a, b\}$, $\mathrm{Back}\, S = \{c\}$ *and* $P = \mathrm{In}\, S$.

1. *If σ is a $(k-1)$-triangle, it is linked only to the edge ω given by $P \cup \{a\}$ and $P \cup \{b\}$.*
2. *If σ is a $(k-2)$-triangle, it is linked to the vertex $\gamma = P \cup \{a, b\}$. Moreover, it is also linked to the edges $\omega_1, \omega_2 \in [\gamma, \sigma]$ and that is all it is linked to.*

Proof. A k-Delaunay triangle σ corresponds to a k-Voronoi vertex, the circumcenter of $a, b, c \in X$. Let $S_\cdot$ and $\mathrm{meb}(\cdot)$ be the k-Delaunay circle and the minimal enclosing ball of a set of sites. We argue geometrically to show (1.), using Figure 4 as reference. Item (2.) is similar.

The center of S_ω must be at the intersection of the k-Voronoi regions for $P \cup \{a\}$ and $P \cup \{b\}$, the red edge l_3. Moreover, a and b must be on S_ω, hence the minimum radius is reached at the end of l_3 that is exactly z, the center of S_σ. Therefore $S_\omega = S_\sigma$.

Let V_σ, V_ω be the set of all involved sites in σ and ω. Recall that a MEB is the smallest circumcircle of its boundary sites. Hence, if we show that $\mathrm{meb}(V_\omega)$ and $\mathrm{meb}(V_\sigma)$ have same boundary sites, they must be the same. Furthermore, V_σ and V_ω differ only by c. Therefore, it is sufficient to show that $\mathrm{meb}(V_\sigma)$ does not contain c on its boundary. Assume $c \in \mathrm{On}\, \mathrm{meb}(V_\sigma)$. Then z is in the closed half-planes $\mathcal{H}_{ac}$ and $\mathcal{H}_{bc}$, where $\mathcal{H}_{\cdot c}$ is the half-plane of all points closer to $\cdot$ than to c. Then the intersection of $\mathcal{H}_{ac}$ and $\mathcal{H}_{bc}$ has z as the closest point to c, meaning that z is also the center of the Delaunay circle, a contradiction to the fact that σ is bad. Hence, $s_{\check{C}ech}(\sigma) = s_{\check{C}ech}(\omega)$ and ω is linked to σ.

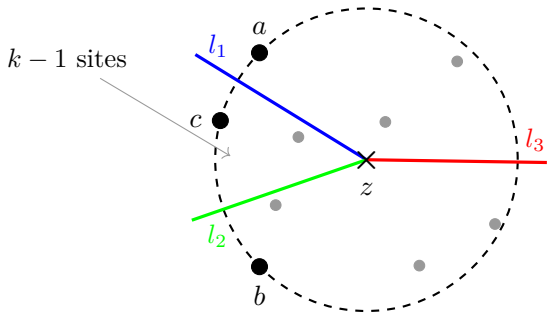


Figure 4 A bad $(k-1)$-triangle. The edges are k-Voronoi edges.

Finally, any other edge that is a face of σ does not have both a and b as involved sites and therefore has different Delaunay circle: for the edge defined by P with b and c, the Delaunay circle must exclude a and its center is on the green line l_2, translated from z in the direction of the midpoint between b and c. The minimum radius is reached either at the midpoint, or at some critical point before. For the edge defined by P with a and c, the argument is the same using the blue line l_1. Hence these edges and their vertices have different Delaunay circle and are not linked to ω. ◀

Lemma 3.2 inspires the construction of an interval partition on $\mathrm{Del}^k(X)$: we put every good simplex into a singleton interval, every bad $(k-1)$-triangle into a pair {edge, triangle} and every bad $(k-2)$-triangle into a diamond {vertex, edge, edge, triangle} of adjacent simplices, as in Lemma 3.2. Hence all simplices in the same interval are linked. Moreover, every bad simplex is linked to a triangle (we omit the proof for the lack of space). Hence:

▶ **Theorem 3.3.** *The above defined intervals partition $\mathrm{Del}^k(X)$. Moreover, bad simplices are in non-singular intervals.*

As a consequence, we have our main result.

▶ **Theorem 3.4.** *Let $X \subseteq \mathbb{R}^2$ be finite and in general position and $k \geq 1$. For every $r \geq 0$, $\mathrm{DelČech}^k_r(X) \searrow \mathrm{Del}^k_r(X)$.*

Proof. For $s \geq r$, consider the set of simplices with Čech radius at most r and Delaunay radius at most s :

$$A(s) \stackrel{def}{=} \{\sigma \in \mathrm{DelČech}^k_r(X) \mid s_{Del}(\sigma) \leq s\}.$$

$A(s)$ is a simplicial complex, because it is the intersection of $\mathrm{DelČech}^k_r(X)$ and $Del^k_s(X)$. Moreover, $A(r) = \mathrm{Del}^k_r(X)$ and $A(\infty) = \mathrm{DelČech}^k_r(X)$. Since X is finite, s_{Del} has finitely many critical values $s_1, s_2, \cdots, s_m$, with $s_1 = r$, and thus there is a filtration

$$\mathrm{Del}^k_r(X) = A_1 \subset A_2 \subset \cdots \subset A_m = \mathrm{DelČech}^k_r(X), \text{ where } A_i = A(s_i).$$

We show that A_i arises from A_{i+1} via free collapses. Note that

$$A_{i+1} \setminus A_i = \{\sigma \in \mathrm{Del}^k(X) \mid s_{Del}(\sigma) = s_{i+1}, s_{\check{C}ech}(\sigma) \leq r\}$$

and these are bad simplices. By Theorem 3.3, these simplices must form one pair or one diamond interval. For a diamond interval, denote the involved simplices v, e_1, e_2 and t. t is maximal, as any triangle in a triangulation, and e_1 and e_2 are both free in A_{i+1}, because removing the diamond interval yields the simplicial complex A_i. Analogously, v is only incident to e_1 and e_2. Hence, we can free-collapse (e_1, t) and afterwards (v, e_2) to obtain A_i.

The case of a pair interval is analogous. Concatenating these free collapses for every $i = 1, \cdots, m-1$, we obtain that $\mathrm{DelČech}^k_r(X) \searrow \mathrm{Del}^k_r(X)$. ◀

4 Conclusion

We constructed, for a fixed radius, a sequence of simplicial collapses from the higher order Delaunay-Čech to the higher order Delaunay complex of a finite set of points in general position in the plane. Our approach does not directly generalize for higher dimensions because we rely on the restriction that the back set has at most one site, while in $\mathbb{R}^d$ the back set can have up to $d-1$ sites. Another difficulty is that the kth order Delaunay complex can have simplices of higher dimension than the ambient space when the point set is not planar. Even so, we do hope that our methods can be adapted to not only show the collapse for higher dimensions, but also to show the existence of a collapsing sequence from the higher order Čech complex to the Delaunay-Čech one.

References

1 Franz Aurenhammer, Rolf Klein, and Der-Tsai Lee. *Voronoi diagrams and Delaunay triangulations*. World Scientific Publishing Co. Pte. Ltd., Hackensack, NJ, 2013. doi:10.1142/8685.

2 Ulrich Bauer and Herbert Edelsbrunner. The Morse theory of Čech and Delaunay complexes. *Trans. Amer. Math. Soc.*, 369(5):3741–3762, 2017. doi:10.1090/tran/6991.

3 Andrew J. Blumberg and Michael Lesnick. Stability of 2-parameter persistent homology. *CoRR*, abs/2010.09628, 2020. URL: https://arxiv.org/abs/2010.09628, arXiv:2010.09628.

4 Herbert Edelsbrunner and John L. Harer. *Computational topology*. American Mathematical Society, Providence, RI, 2010. An introduction. doi:10.1090/mbk/069.

5 Herbert Edelsbrunner and Georg Osang. A simple algorithm for higher-order Delaunay mosaics and alpha shapes. *CoRR*, abs/2011.03617, 2020. URL: https://arxiv.org/abs/2011.03617, arXiv:2011.03617.

6 Steven Fortune. Voronoi diagrams and Delaunay triangulations. In Jacob E. Goodman and Joseph O'Rourke, editors, *Handbook of Discrete and Computational Geometry, Second Edition*, pages 513–528. Chapman and Hall/CRC, 2004. doi:10.1201/9781420035315.ch23.

Nearest-Neighbor Decompositions of Drawings*

Jonas Cleve[†1], Nicolas Grelier[‡2], Kristin Knorr[§1], Maarten Löffler[3], Wolfgang Mulzer[¶1], and Daniel Perz[‖4]

1 **Freie Universität Berlin**
{jonascleve, knorrkri, mulzer}@inf.fu-berlin.de
2 **ETH Zürich, Department of Computer Science**
nicolas.grelier@inf.ethz.ch
3 **Utrecht University**
m.loffler@uu.nl
4 **TU Graz**
daperz@ist.tugraz.at

Abstract

Let $\mathcal{D}$ be a straight-line drawing of a graph in the plane, and let c be a positive integer. We say that $\mathcal{D}$ has a nearest-neighbor decomposition into c parts if there are point sets $P_1, \ldots, P_c$ such that $\mathcal{D}$ is the union of the nearest neighbor graphs on $P_1, \ldots, P_c$. We study the complexity of deciding whether it is possible to draw D as the union of c nearest-neighbor graphs.

1 Introduction

Let $P \subset \mathbb{R}^2$ be a finite planar point set, and let C be a finite set of *colors*. A *coloring* is a function $\sigma : P \to C$ that assigns a color to each point in P. For any color $c \in C$, we write $P_c = \{p \in P \mid \sigma(p) = c\}$ for the points in P that were colored with c.

The *nearest-neighbor graph for a color* $c \in C$, $\mathbb{N}_c$, is the embedded graph with vertex set P_c and a straight-line edge between $p, q \in P_c$ if and only if p is the nearest neighbor of q among all points in P_c, or vice versa.[1] We will consider $\mathbb{N}_c$ both as a combinatorial graph, consisting of vertices and edges, and as a subset of the plane, consisting of the points in P_c and the line segments that represent the edges. We write $\mathbb{N} = \bigcup_{c \in C} \mathbb{N}_c$ for the union of the nearest-neighbor graphs of all colors. Again, we consider $\mathbb{N}$ both as a graph and as a set.

We are interested in the following problem: suppose we are given a *drawing* $\mathcal{D}$, i.e., a graph that is embedded in the plane and whose edges are represented by (possibly crossing) straight line segments. Our general task is to construct a point set P, a set of colors C, and a color assignment σ, such that the union $\mathbb{N}$ of the nearest-neighbor graphs for P and C equals $\mathcal{D}$, considered as subsets of the plane. We call $\mathbb{N}$ an *NN-decomposition* of $\mathcal{D}$ with vertex set P, and we call $|C|$ the *color-number* of $\mathbb{N}$. See Figure 1.

* This research was started at the 4th DACH Workshop on Arrangements and Drawings, February 24–28, 2020, in Malchow, Germany. We thank all participants of the workshops for valuable discussions and for creating a conducive research atmosphere.
† Supported in part by ERC StG 757609.
‡ Supported by the Swiss National Science Foundation within the collaborative DACH project *Arrangements and Drawings* as SNSF Project 200021E-171681.
§ Supported by the German Science Foundation within the research training group 'Facets of Complexity' (GRK 2434).
¶ Supported in part by ERC StG 757609 and by the German Research Foundation within the collaborative DACH project *Arrangements and Drawings* as DFG Project MU 3501/3-1.
‖ Partially supported by FWF within the collaborative DACH project *Arrangements and Drawings* as FWF project I 3340-N35

[1] We assume general position but our notion is different: Three points can be on a line but we require that the pairwise distances between the points are all different. Note that our notion of nearest-neighbor graph is undirected, but a directed version also exists.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

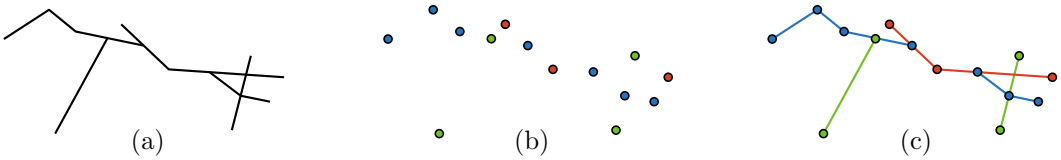


Figure 1 (a) A drawing. (b) A set of points with 3 colors. (c) The 3 nearest-neighbor graphs.

Related work. Our problem is motivated from automated content generation for puzzle games: van Kapel introduces a version of *connect-the-dots* puzzles in which the task is to connect dots based on colors rather than numbers [7]. In this puzzle, points may have multiple colors; see Figure 2. He implemented a heuristic approach for generating such puzzles which works well for small instances, but for larger instances generates too many colors to be practical [5].

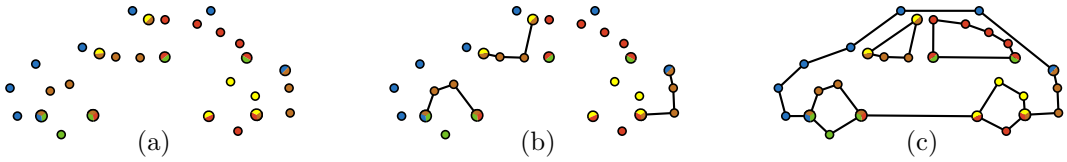


Figure 2 (a) Multi-colored points with 5 colors: blue, red, green, yellow, and orange. (b) The orange nearest-neighbor graph. (c) The union of all nearest-neighbor graphs. Picture taken from [5].

The nearest-neighbor graph of a set of points in the plane is well understood [1, 6]. It is a subgraph of the relative neighborhood graph of the points [4, 6], which in turn is a subgraph of the Delaunay triangulation. The problem of recognizing whether a given abstract graph can be realized as a nearest-neighbor graph of a point set is open and conjectured to be hard. In contrast, testing whether a given embedded graph is a (single) nearest-neighbor graph is easy.

Results. We show that testing if a drawing can be decomposed into c nearest-neighbor graphs so that every vertex of the drawing is a vertex of at least one of the graphs, is polynomial for $c \leq 2$, but NP-hard for $c \geq 3$. If we allow edges of the nearest-neighbor graphs to cross without requiring a vertex at the crossing, the problem is already hard for $c = 2$.

2 Existence of NN-Decompositions on Special Points

Let $\mathcal{D}$ be a straight-line drawing. The *segments* of $\mathcal{D}$ are the inclusion maximal line segments in $\mathcal{D}$. If s is a line segment with $s \subset \mathcal{D}$ such that s is not a segment of $\mathcal{D}$, we say that s is *covered* by $\mathcal{D}$. The *special* points of $\mathcal{D}$ are the endpoints of the line segments in $\mathcal{D}$ and the intersection points between two segments in $\mathcal{D}$. We require a stronger general position assumption, namely that the pairwise distances between the special points of $\mathcal{D}$ are all distinct. We consider the special case of our problem where the vertex set of the NN-decomposition must consist of the special points. We investigate the question under which circumstances it is possible to find such a *special-point NN-decomposition* of $\mathcal{D}$.

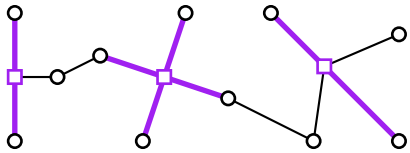


Figure 3 This drawing highlights the possible violations that make a drawing non-plane.

2.1 The Plane Case

A drawing $\mathcal{D}$ is called *plane* if its segments intersect only at their endpoints, i.e., no segment of $\mathcal{D}$ contains a special point in its relative interior; see Figure 3 for an illustration.

▶ **Lemma 2.1.** *Let $\mathcal{D}$ be a plane drawing. Suppose there is a special-point NN-decomposition $\mathbb{N}$ of $\mathcal{D}$. Let σ be the underlying coloring of $\mathbb{N}$. Then, for any connected component $\mathcal{C}$ of $\mathcal{D}$, the coloring σ assigns the same color to all special points in $\mathcal{C}$.*

Proof. Suppose $\mathcal{D}$ has a connected component $\mathcal{C}$ in which σ assigns two distinct colors. Then, $\mathcal{C}$ contains a segment $s = uv$ whose endpoints are colored differently. However, the line segment uv must be covered by $\mathbb{N}$, and thus, there exists a segment t in $\mathbb{N}$ that contains u, v, and another special point of $\mathcal{D}$ (since the segments in $\mathbb{N}$ are derived from nearest-neighbor relations between points of the same color). By our general position assumption, the segment t is not in $\mathcal{D}$, so $\mathbb{N}$ is not an NN-decomposition of $\mathcal{D}$, a contradiction. ◀

Let $\mathcal{C}$ be a connected component in a plane drawing $\mathcal{D}$, p a special point in $\mathcal{C}$. We denote by $a(p)$ the special point in $\mathcal{C} \setminus \{p\}$ that is closest to p. We denote by $b(p)$ the set of special points in $\mathcal{D}$ that are strictly closer to p than $a(p)$. By definition, $b(p) \subset \mathcal{D} \setminus \mathcal{C}$. Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two distinct connected components. We say that $\mathcal{C}_1$ and $\mathcal{C}_2$ are *incompatible* if there is a special point $p \in \mathcal{C}_1$ such that $b(p) \cap \mathcal{C}_2 \neq \emptyset$, or vice-versa.

We call the connected component $\mathcal{C}$ *NN-representable* if $\mathcal{C}$ is the nearest-neighbor graph of its special points. By Lemma 2.1, if $\mathcal{C}$ is plane, then NN-representability is equivalent to special-point NN-decomposability.

Let $\mathcal{D}$ be a straight-line drawing. We denote by $V = \{\mathcal{C}_i\}_{1 \leq i \leq n}$ the connected components of $\mathcal{D}$. We define E as the set of pairs $\{\mathcal{C}_i, \mathcal{C}_j\}$ where $\mathcal{C}_i$ and $\mathcal{C}_j$ are incompatible. We say that the graph $G := (V, E)$ is the *incompatibility graph* of $\mathcal{D}$.

▶ **Theorem 2.2.** *Let C be a set of colors with $|C| \leq 2$. There is a polynomial time algorithm for the following task: given a plane drawing $\mathcal{D}$, is there a special-point NN-decomposition of $\mathcal{D}$ with color set C?*

Proof. Let $\mathcal{D}$ be a plane drawing. By Lemma 2.1 each connected component is colored with one color if there is a special-point NN-decomposition of $\mathcal{D}$. This means every connected component of $\mathcal{D}$ is NN-representable if there is a special-point NN-decomposition of $\mathcal{D}$. This can be checked in polynomial time, as we only need to compute the nearest-neighbor graph of the special vertices. If the test is negative, the algorithm answers that there is no solution.

Otherwise, we construct the incompatibility graph G of $\mathcal{D}$, and we check whether G can be colored with C. This takes polynomial time since $|C| \leq 2$ (for $|C| = 2$ check whether G is bipartite, for $|C| = 1$ check that G has no edges). Note, that if G was not C-colorable, then there exists at least one incompatible pair $\{\mathcal{C}_i, \mathcal{C}_j\}$ which was assigned the same color. Since by definition $b(p) \cap \mathcal{C}_j \neq \emptyset$ for a $p \in \mathcal{C}_i$, such a coloring would create a non-existing edge between $\mathcal{C}_i$ and $\mathcal{C}_j$. If possible, we give all special points in a component $\mathcal{C}$ the color assigned to the corresponding vertex in G. Since all connected components are NN-representable, this

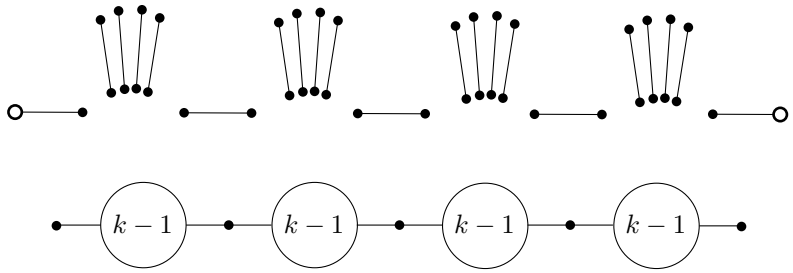


Figure 4 A 5-wire of length 5 and the general incompatibility graph of a k-wire of length 5.

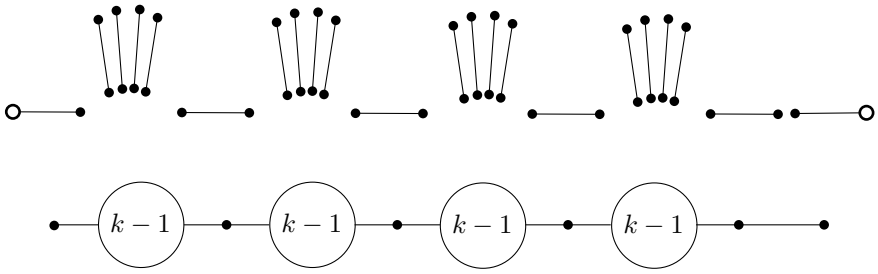


Figure 5 A 5-chain of length 5 and the general incompatibility graph of a k-chain of length 5.

is also a special point NN-decomposition of $\mathcal{D}$. If G cannot be colored with C, then there is no special point NN-decomposition of $\mathcal{D}$. Assume to the contrary that there is a special point NN-decomposition of $\mathcal{D}$. Since every point of a connected component has the same color, we color the corresponding vertex in G with the same color. But then we would have a coloring of G, which contradicts that G cannot be colored with C. ◀

▶ **Theorem 2.3.** *Let C be a set of colors with $|C| \geq 3$. The following task is NP-complete: given a plane drawing $\mathcal{D}$, is there a special-point NN-decomposition of $\mathcal{D}$ with color set C?*

Proof. Gräf, Stumpf, and Weißenfels [3] showed how to reduce k-colorability to k-colorability of unit disk graphs. Our proof is inspired by theirs. Let $k = |C|$. We show the NP-hardness of coloring the special points of $\mathcal{D}$ with $k \geq 3$ colors by means of a reduction from k-colorability. We make use of four types of gadgets: k-wires, k-chains, k-clones, and k-crossings. They are depicted in Figures 4 to 7, together with their incompatibility graphs. The symbol consisting of a number x in a circle denotes a clique of size x. A vertex v connected to such a symbol means that there is an edge between v and all the vertices of the clique. These incompatiblity graphs are exactly the gadgets defined by Gräf, Stumpf, and Weißenfels. Note that each connected component in these gadgets is NN-representable. The gadgets shown are for $k = 5$.

In Figures 4 to 6, there are several sets of four segments that are very close and nearly vertical. For other values of k, the gadgets are analogous, but with $k - 1$ almost vertical segments instead of four. Similarly, in Figure 7, there are five sets consisting of three close segments. For other values of k, there are five sets of $k - 2$ segments. In Figures 4 and 5, k-wires and k-chains are drawn as if they were on a line, but they may also bend with a right angle. Note that in Figures 4 to 7 some vertices are specially marked with larger empty circles. These vertices will be called *extreme vertices*.

In Figure 7, there seem to be points lying on a segment between two other points. Actually, these points are shifted by a sufficiently small $\varepsilon > 0$, to ensure our general position assumption.

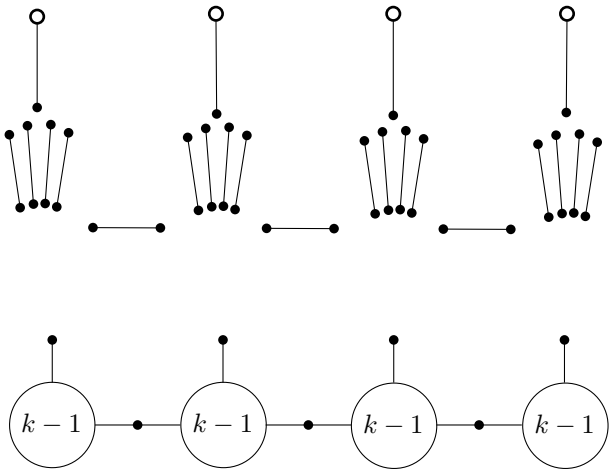


Figure 6 A 5-clone of length 4 and and the general incompatibility graph of a k-clone of length 4.

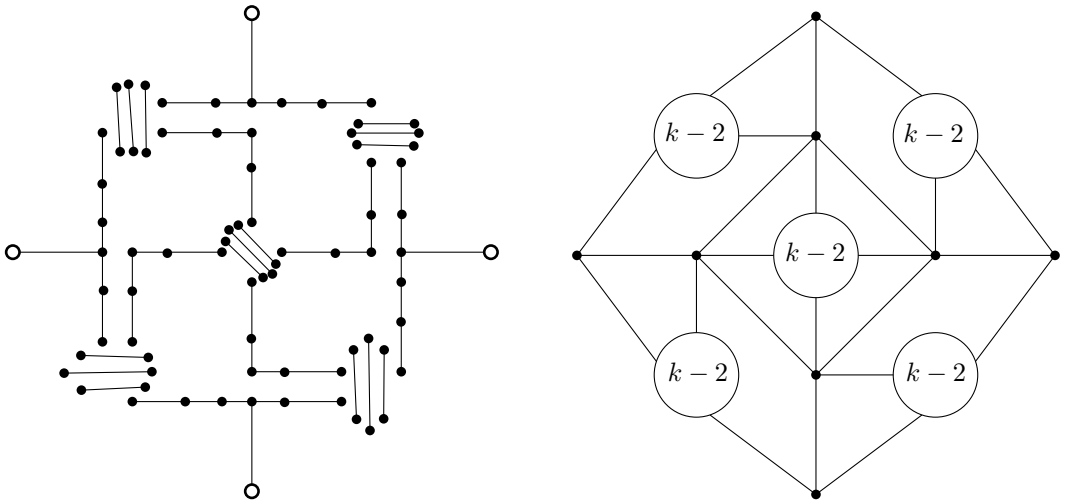


Figure 7 The 5-crossing gadget (left); the incompatibility graph of the k-crossing gadget (right).

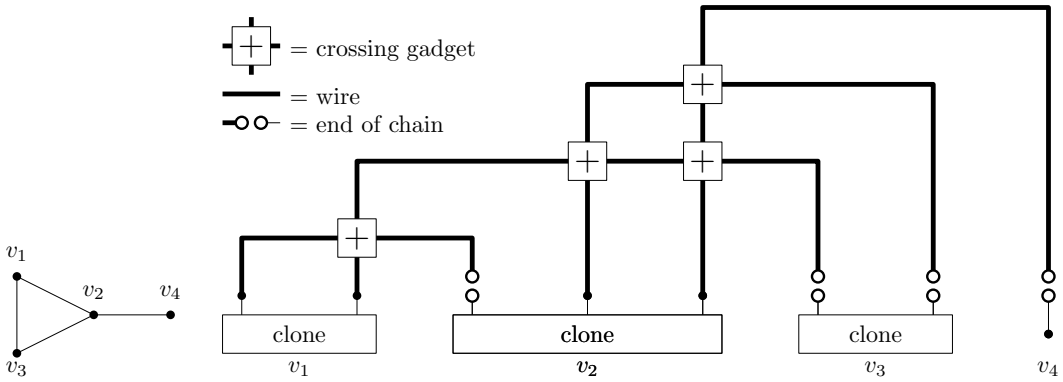


Figure 8 An example graph G with four vertices (left). Converting G to an NN-graph (right).

The main property of a k-wire is that in any coloring with k colors of its incompatibility graph, the extreme vertices are assigned the same color. In contrast, in a k-chain, the extreme vertices are assigned different colors. In a k-clone of length ℓ, there are ℓ extreme vertices. In any coloring with colors from C, all extreme vertices have the same color. Finally, for the k-crossing, opposite extreme vertices must have the same color; a pair of consecutive extreme vertices (e.g., top and left extreme vertices) may or may not be assigned the same color [2].

Now we follow the proof of Gräf, Stumpf, and Weißenfels. Suppose we are given a graph $G = (V, E)$. We describe a drawing $\mathcal{D}$ whose incompatibility graph can be colored with color set C if and only if the vertices of G can be colored with C. Refer to Figure 8. For each vertex v of degree δ in G, we draw a k-clone of size δ. The clones are drawn so that they are arranged on a horizontal line and such that their upper points have the same y-coordinate. Then, for each edge $\{u, v\} \in E$, we draw it on the plane as two vertical segments, each incident to one k-clone, and one horizontal segment that connects the two upper points of the vertical segments. We do that such that for any pair of edges, their horizontal segments have distinct y-coordinates. Then we replace each crossing between a pair of edges by a k-crossing. Finally, let us consider one edge $\{u, v\} \in E$, and let us orient it arbitrarily, say toward v. We replace each part of the edge between two k-crossings by k-wires of sufficient length. If there are no crossings, we replace the edge by a k-chain. Otherwise, the part of the edge between u and the first k-crossing is replaced by a k-wire, and the part between the last k-crossing and v is replaced by a chain. As the points of distinct gadgets are sufficiently remote (except for pairs of gadgets that are connected on purpose), the incompatibility graph of this drawing is the union of the incompatibility graphs of the individual gadgets.

It is possible to find positions with a polynomial number of bits such that all pairwise distances are distinct but at the same time the positions are sufficiently close to the prescribed positions. This concludes the reduction. ◀

2.2 The non-plane case

We show that if drawings are not required to be plane, the problem is hard for two colors.

▶ **Theorem 2.4.** *Let C be a set of colors with $|C| = 2$. The following task is NP-complete: given a drawing $\mathcal{D}$, is there a special-point NN-decomposition of $\mathcal{D}$ with color set C?*

Proof. We reduce from Not-All-Equal 3SAT (NAE-3SAT). Let Φ be an NAE-3SAT formula with variable set X and clause set Y. Let G_Φ be the associated bipartite graph with vertex set $X \cup Y$, where two vertices x and y are adjacent if and only if x is a variable that appears

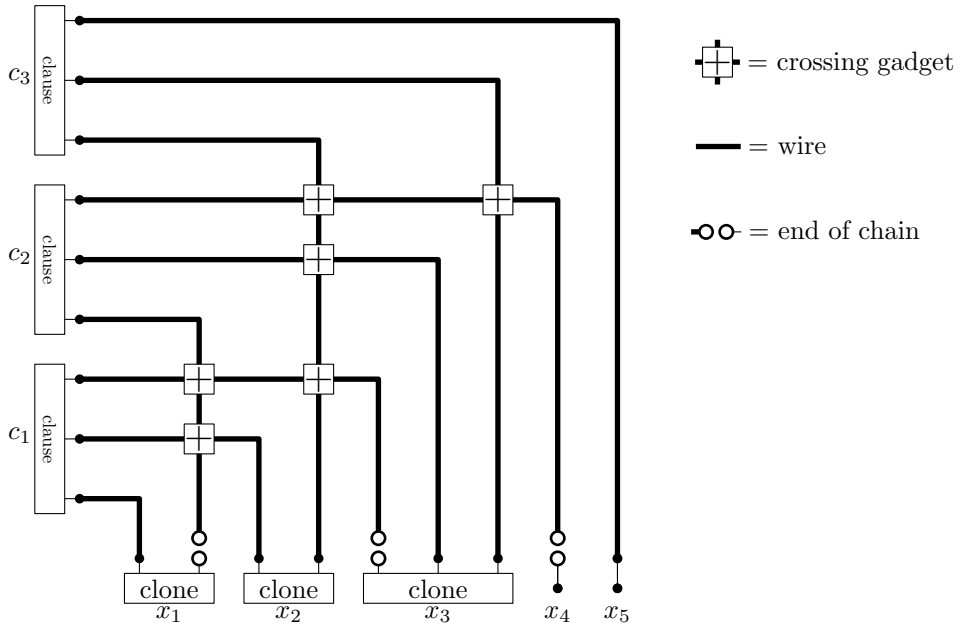


Figure 9 Structure of the conversion of the NAE-3SAT formula with clauses $c_1 = (x_1, x_2, \neg x_3)$, $c_2 = (\neg x_1, x_3, \neg x_4)$, and $c_3 = (x_2, x_3, x_5)$ into a 2-color $\mathbb{N}$ graph.

in clause y. We draw G_Φ as follows: clauses are represented by vertical segments on the y-axis of length 3. Variables of degree δ are represented as horizontal segments on the x-axis of length δ. Each edge $\{x, y\}$ is drawn as the union of one vertical and one horizontal segment. The vertical segment is incident to the variable gadget for x. The horizontal segment is adjacent to a clause gadget for y. See Figure 9 for an example.

We use some gadgets from the proof of Theorem 2.3. We replace each variable by a 2-clone of length δ. We replace each clause by the gadget in Figure 10 (see Figure 11 for assignments where all literals have the same color), and each crossing by the gadget in Figure 12. In Figure 12, some points have been colored. Note that this does not correspond to an assignment of truth values, but is supposed to provide visual information for the reader. The distance between a green point and a blue point is $1 - \varepsilon$, for a sufficiently small $\varepsilon > 0$. The distance between a blue point and the red point is 1. The distance between the red point and an orange point is $1 + \varepsilon$. The blue point on the left and the orange point on the right are finally shifted by a suitable $\eta > 0$ with $\eta \ll \varepsilon$, so that no two points are at the same distance from the red point. The points in the clause gadget that are on the vertical connected component on the left side are arranged so that this connected component is NN-representable. Finally, each part of an edge between two gadgets is replaced by a 2-wire of suitable length. We have thus obtained a drawing $\mathcal{D}$.

We claim that Φ is satisfiable if and only if there exists a special-point NN-decomposition of $\mathcal{D}$ with two colors. First, notice a clause gadget has a special-point NN-decomposition if and only if two of the horizontal segments on the right side are assigned different colors. In the non-plane crossing gadget opposite segments are assigned the same color. All of them may be assigned the same color, as in Figure 13a, or consecutive segments might be assigned different colors, as in Figure 13b. Therefore, by associating the colors of C with truth values, $\mathcal{D}$ has a special-point NN-decomposition if and only if Φ is satisfiable. ◀

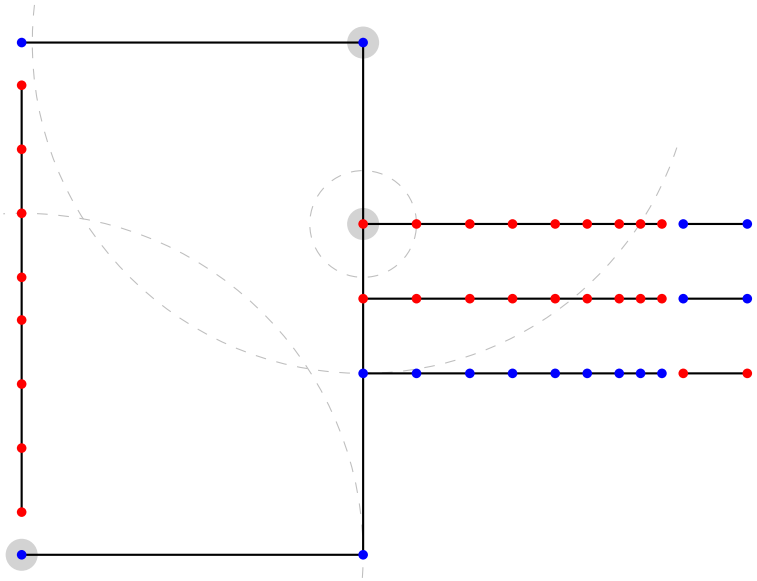
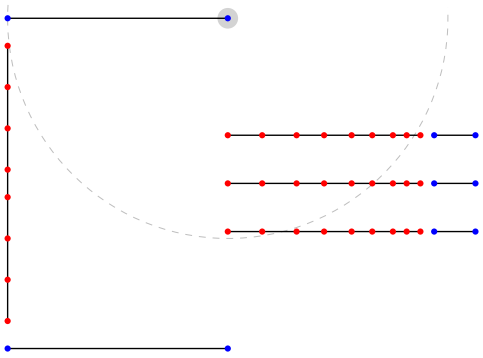
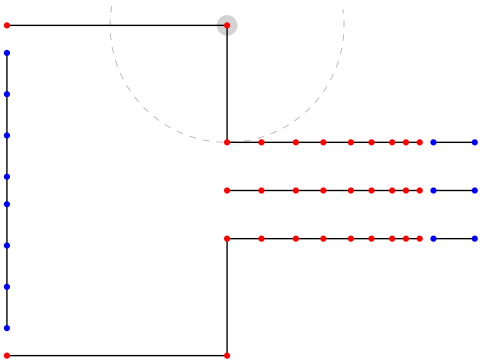


Figure 10 A clause gadget with a valid assignment. For the highlighted vertices the dashed circle indicates the distance to the nearest neighbor of the same color.



(a) The top and bottom right vertices connect to the vertices to their left.



(b) The vertical connection from the marked vertex does not span the full height.

Figure 11 Invalid assignments on the non-plane clause gadget where all three incoming wires have the same color.

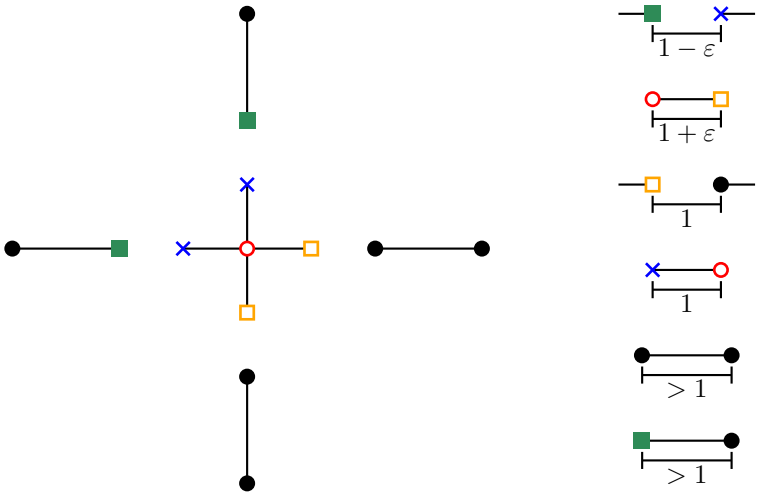


Figure 12 A non-plane crossing gadget.

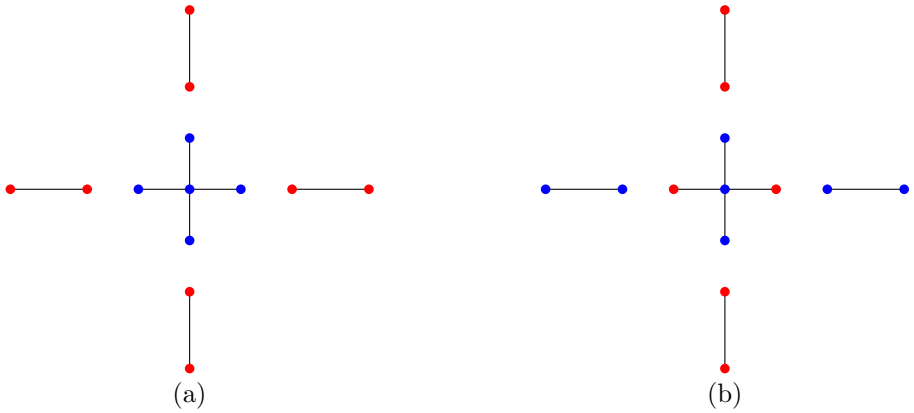


Figure 13 Two valid assignments on the non-plane crossing gadget. (a) All extreme segments are assigned the same color. (b) Opposite segments are assigned the same color.

References

1 D. Eppstein, M.S. Paterson, and F.F. Yao. On nearest-neighbor graphs. *Discrete Comput Geom*, 17:263–282, 1997. doi:10.1007/PL00009293.

2 Michael R. Garey, David S. Johnson, and Larry J. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1:237–267. doi:10/dwvqpj.

3 Albert Gräf, Martin Stumpf, and Gerhard Weißenfels. On coloring unit disk graphs. *Algorithmica*, 20(3):277–293, 1998.

4 J.W. Jaromczyk and G.T. Toussaint. Relative neighborhood graphs and their relatives. *Proc. of the IEEE*, 80(9):1502–1517, 1992.

5 Maarten Löffler, Mira Kaiser, Tim van Kapel, Gerwin Klappe, Marc van Kreveld, and Frank Staals. The connect-the-dots family of puzzles: Design and automatic generation. *ACM Transactions on Graphics*, 33(4):72, 2014. doi:10.1145/2601097.2601224.

6 Joseph S. B. Mitchell and Wolfgang Mulzer. Proximity algorithms. In Jacob E. Goodman, Joseph O'Rourke, and Csaba D. Tóth, editors, *Handbook of Discrete and Computational Geometry*, chapter 32, pages 849–874. CRC Press, Boca Raton, 3rd edition, 2017. doi:10.1201/9781315119601.

7 Tim van Kapel. Connect the closest dot puzzles. Master's thesis, Utrecht University, 2014. URL: http://dspace.library.uu.nl/handle/1874/296600.

Hardness of Recognition and 3-Coloring of L-graphs

Petr Chmel[*1] **and Vít Jelínek**[†2]

1 **Computer Science Institute, Charles University**
chmel@kam.mff.cuni.cz
2 **Computer Science Institute, Charles University**
jelinek@iuuk.mff.cuni.cz

Abstract

This paper focuses on intersection graphs of axis-aligned L-shapes, so-called L-graphs, and a similar class of intersection graphs of axis-aligned L-shapes and J-shapes, known as {L, J}-graphs; in both cases, the vertices of a graph are represented by curves, with two vertices sharing an edge if and only if their respective curves intersect. We show that recognition of both L-graphs and {L, J}-graphs is NP-complete, answering a question of Felsner, Knauer, Mertzios and Ueckerdt. Moreover, we show that deciding if a triangle-free L-graph is 3-colorable is also NP-complete.

1 Introduction

A problem that is NP-complete for the class of all graphs may be solvable in polynomial time for a restricted class of graphs. However, the recognition of graphs in a restricted class may itself be an NP-complete problem. We will focus on the restriction of decision problems to intersection graphs of piecewise linear axis-aligned curves. Chaplick et al. [3] showed that for each $k \geq 1$, the class of intersection graphs of piecewise linear axis-aligned curves with at most k bends is NP-complete to recognize. This class is usually denoted by B_k-VPG where VPG stands for Vertex intersection graphs of Paths in a Grid. A similar result was obtained by Kratochvíl [10], who showed that recognition of grid intersection graphs (intersection graphs of horizontal and vertical line segments with the only intersections being between a vertical and a horizontal segment) is NP-complete as well.

The class B_k-VPG has been in the spotlight since Asinowski et al. [1] showed in 2012 that the class of planar graphs is contained in B_3-VPG. Chaplick and Ueckerdt [4] showed a year later that in fact two bends suffice. Biedl and Derka [2] strengthened this result by showing that not only are two bends sufficient, but we may also require that any two curves intersect at most once. Finally, Gonçalves, Isenmann, and Pennarun [7] showed in 2018 that a single bend is sufficient and, furthermore, only curves oriented as the letter L are needed.

Thus, while the class B_1-VPG permits any of the four shapes L, J, Γ, ⅂, only one of these shapes is needed when representing planar graphs. The intersection graphs of just one of these L-shapes are called L-graphs. Apart from planar graphs, L-graphs also include circle graphs [1] (the intersection graphs of chords in a circle) and, by definition, B_0-VPG.

The inclusion of planar graphs in the class of L-graphs motivates a number of complexity questions, two of which we shall focus on in this paper. First, it has long been known [6] that the decision problem of 3-coloring planar graphs is NP-complete, while the decision problem of 3-coloring triangle-free planar graphs is in P by Grötzsch's theorem [8]. However,

* Supported by the grant no. 21-32817S of the Czech Science Foundation (GAČR).
† Supported by the grant no. 18-19158S of the Czech Science Foundation (GAČR).

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

the complexity of 3-coloring triangle-free L-graphs is unknown. Second, while planar graphs are recognizable in polynomial time, the complexity of recognition of L-graphs has not yet been established.

We show that both problems – 3-coloring of triangle-free L-graphs and recognition of L-graphs – are NP-complete. Furthermore, we show that the recognition of $\{\mathsf{L}, \mathsf{J}\}$-graphs is NP-complete as well, answering an open problem of Felsner et al. [5].

1.1 Basic definitions

A *representation* R of a finite graph $G = (V, E)$ is a family of piecewise linear curves in the plane, $R = \{R(v) \colon v \in V\}$, such that $R(v) \cap R(u)$ is non-empty if and only if $\{u, v\} \in E$. A representation is *proper* if every curve is simple, there are finitely many intersection points and finitely many bends, and in every point of the plane, at most two curves intersect and every such intersection is a crossing. An L-*shape* is the union of a horizontal line segment and a vertical line segment such that the only common point of the two line segments is the lowest point of the vertical line segment and the leftmost point of the horizontal line segment. It may also happen that the line segments consist of a single point. An L-*representation* is a representation such that every curve is an L-shape. An L-*graph* is a graph that admits a proper L-representation.

2 Recognition

To show that the recognition of L-graphs is NP-complete, we adapt the method used by Chaplick et al. [3] to show that recognition of graphs in B_1-VPG is NP-complete.

As our main tool, we use the Noodle-Forcing Lemma of Chaplick et al. [3]. When specialized to our situation, the Noodle-Forcing Lemma shows that for any proper L-representation R of a graph G, there is a graph G' containing G as an induced subgraph, such that in any proper $\{\mathsf{L}, \mathsf{J}\}$-representation R' of G', the curves representing the vertices of G have the same order of intersection points as in R, up to possible reversal. The representation R' is obtained by overlaying the representation R by a sufficiently dense grid-like configuration of axis-aligned segments, as illustrated in Figures 1 and 2.

We apply the Noodle-Forcing Lemma to the L-representation of K_5 from Figure 2. This forces the order of intersection points on the L-shapes, which in turn by Lemma 1 below forces a bend for the middle L-shape. The forced bend then allows us to create a gadget that behaves similarly to a line segment. This allows us to reduce from the recognition of grid intersection graphs which Kratochvíl showed to be NP-complete [10].

▶ **Lemma 1.** *For a proper* $\{\mathsf{L}, \mathsf{J}\}$*-representation* R *of* $K_5, V(K_5) = \{a, b, c, d, e\}$*, if the intersection points of all five vertices are located in either alphabetical or reverse alphabetical order on each curve, then the vertex* c *cannot be represented just as a horizontal or a vertical line segment.*

The proof of the lemma is omitted.

We can now state and prove the main result.

▶ **Theorem 2.** *The recognition of* L*-graphs and the recognition of* $\{\mathsf{L}, \mathsf{J}\}$*-graphs are both NP-complete.*

Proof. It is easy to see that the two problems are in NP. We show that they are NP-hard.

We reduce from the recognition problem of grid intersection graphs. The argument is similar to the proof of the NP-completeness of B_k-VPG recognition by Chaplick et al. [3],

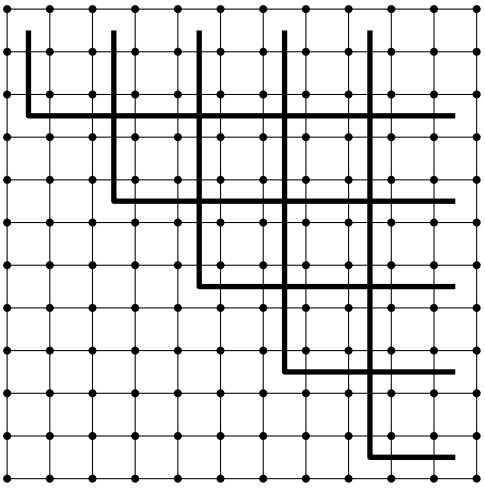


Figure 1 The L-represented graph K_5 with the outline of the grid overlay

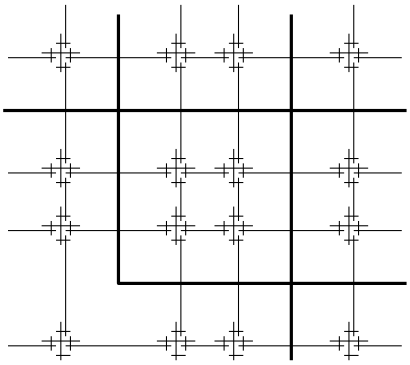


Figure 2 A detail of the grid overlaid on top of the representation of K_5

and we also use the same terminology. Namely, we first construct a *pin*, which we then enhance into a *clothespin*.

In order to construct the pin, we first take a K_5 with vertices a, b, c, d, e, and create its L-representation R so that the intersection points on all five curves are located in alphabetical order. We then apply the Noodle-Forcing Lemma to the representation R, overlaying it with the grid configuration, to obtain a representation R'. We refer to the graph represented by R' as *the pin* $P(c)$, where c is the middle vertex of the L-representation of K_5. We will say that c is the *exposed* vertex of the pin $P(c)$.

As the Noodle-Forcing Lemma forces the order of intersections, we know that in any $\{\mathsf{L}, \lrcorner\}$-representation R'' of $P(c)$, the curve $R''(c)$ is intersected in alphabetic order by $R''(a)$, $R''(b)$, $R''(d)$ and $R''(e)$. Let X be the endpoint of $R''(c)$ that is closer to the intersection with $R''(e)$ than with $R''(a)$. The *tip* of the pin $P(c)$ is the subcurve of $R''(c)$ starting in X and ending with the nearest intersection of $R''(c)$ with another curve of R''.

Lemma 1 guarantees that c has a bend in any $\{\mathsf{L}, \lrcorner\}$-representation of $P(c)$. This means that the tip of $P(c)$ must be a straight line segment, otherwise we could truncate the tip when it reaches the bend, to obtain a representation of K_5 contradicting Lemma 1.

We will now modify the representation R' slightly, so that the tip is in the unbounded face of R', and it may be extended arbitrarily far without crossing any other part of R';

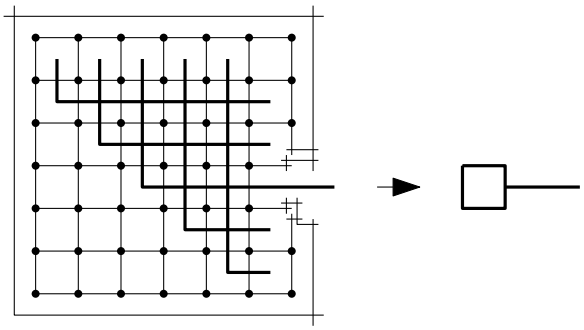


Figure 3 The pin construction

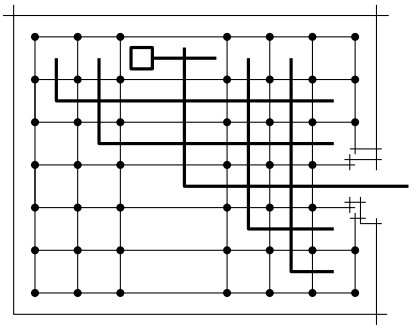


Figure 4 The issue with using pins only

see Figure 3. Note that by flipping the representation R' diagonally, we may create an L-representation of $P(c)$ with a horizontal tip as well as a representation with a vertical tip.

We would like to use the tips of the pins to simulate the behavior of axis-aligned segments in grid intersection graphs. Unfortunately, it may happen that an adjacency of two exposed vertices is not represented by the crossing of the tip of their pins, but by an intersection in another part of the L-shapes, as in Figure 4.

To resolve the issue, we apply an idea of Chaplick et al. [3] and construct a gadget called a clothespin. A *clothespin* $CP(v_1, v_2)$ is the graph whose representation is depicted in Figure 5. It contains two pins $P(v_1)$ and $P(v_2)$. Its key feature is that in any $\{\mathsf{L}, \mathsf{J}\}$-representation R'' of $CP(v_1, v_2)$, there is only one face of R'' that is adjacent to both $R''(v_1)$ and $R''(v_2)$, and this face is only adjacent to the tips of the two curves. In particular, any curve added into R'' that crosses both $R''(v_1)$ and $R''(v_2)$ without crossing any other curve of R'' must cross the tips of the two pins. Moreover, as shown in Figure 5, a clothespin $CP(v_1, v_2)$ has an L-representation in which the tips of the two pins $P(v_1)$ and $P(v_2)$ are parallel segments that can be extended arbitrarily far without crossing the rest of the clothespin. The union of the tips of $P(v_1)$ and $P(v_2)$ is the *tip* of the clothespin $CP(v_1, v_2)$, while the rest of the clothespin is the *head*.

Now, we may describe our reduction. Given a graph G as the input for grid intersection graph recognition, we create a graph G_{L} by replacing every vertex $v \in V(G)$ by a copy $CP(v_1, v_2)$ of the clothespin. If two vertices u, v share an edge of G, then we add to G_{L} four new edges $\{u_i v_j \colon i, j \in \{1, 2\}\}$, where $CP(u_1, u_2)$ and $CP(v_1, v_2)$ are the two clothespins representing u and v.

We now show that if G is a grid intersection graph then G_{L} is an L-graph, while if G is not a grid intersection graph, then G_{L} is not even an $\{\mathsf{L}, \mathsf{J}\}$-graph. This implies the NP-hardness

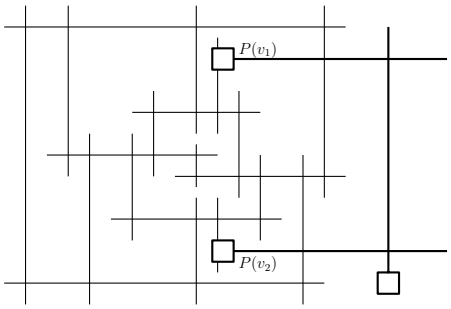


Figure 5 The clothespin construction

of recognition both for L-graphs and for $\{\mathsf{L}, \lrcorner\}$-graphs.

If G has a grid intersection representation, we replace the line segments with clothespins that are thin enough so that their heads are pairwise disjoint and only the tips intersect, thus obtaining an L-representation of G_{L}. Conversely, if G_{L} has an $\{\mathsf{L}, \lrcorner\}$-representation, we take the tip of v_1 in each clothespin as the representation of v, which yields a grid intersection representation of G. ◀

We may show that the recognition of grid intersection graphs remains NP-complete even when an L-representation of the graph is given as part of the input. This is similar to a result of Chaplick et al. [3] who showed, for all $k \in \mathbb{N}_0$, that the recognition of B_k-VPG graphs is still NP-complete even when given a B_{k+1}-VPG representation.

▶ **Theorem 3.** *The recognition of grid intersection graphs is NP-complete even when an* L*-representation of the graph is part of the input.*

Proof. The proof is based on a reduction used by Kratochvíl [10] to show that the recognition of grid intersection graphs is NP-complete. The reduction transforms a 3-SAT formula Φ into a graph $G(\Phi)$ which has a grid intersection representation if and only if Φ is satisfiable.

We only need to argue that for any Φ, the graph $G(\Phi)$ has an L-representation, which can be determined in polynomial time. This also largely follows from Kratochvíl's construction [10], which shows that $G(\Phi)$ has a representation where all vertices are represented by horizontal and vertical segments, except for certain fixed-sized subgraphs, called *clause gadgets*. However, it can be routinely checked that the clause gadgets all have an L-representation (see Figure 6 for an example), yielding an L-representation of $G(\Phi)$. We omit the technical details. ◀

3 3-Coloring

Deciding 3-colorability of planar graphs has long been known to be NP-complete [6], and since all planar graphs are L-graphs [7], 3-colorability of L-graphs is NP-complete as well. However, this argument does not extend to triangle-free graphs, since all planar triangle-free graphs are 3-colorable [8].

Nevertheless, we show that 3-coloring is hard even for triangle-free L-graphs. The first step of our argument is showing that 3-coloring of arbitrary L-graphs is hard, even when an L-representation is given as part of the input. We omit the proof of this result. We remark that the result does not directly follow from the hardness of 3-coloring of planar graphs, because the proof that planar graphs are L-graphs from [7] does not explicitly present a polynomial algorithm to construct an L-representation of a given planar graph.

We now present our second main result.

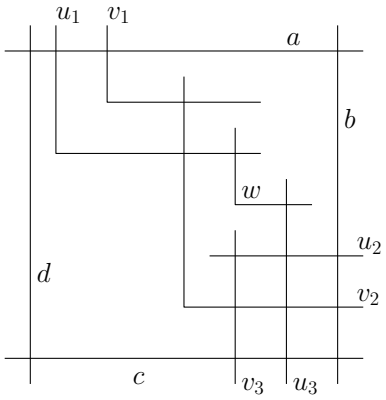


Figure 6 The L-representation of a clause gadget

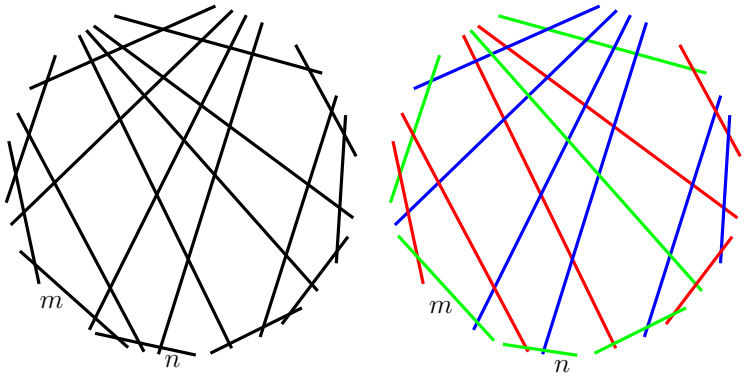


Figure 7 A not 3-colorable circle graph H and its 3-colorable circle subgraph H' created by removing the edge $\{m, n\}$

▶ **Theorem 4.** *The problem of 3-coloring a triangle-free* L*-graph is NP-complete even when the graph's* L*-representation is part of the input.*

Proof. Given an instance of 3-coloring of an arbitrary L-graph G with a given L-representation, we will transform it into an equivalent 3-coloring instance G' of a triangle-free L-graph.

To construct G', we replace every vertex of G with a copy of a triangle-free circle graph H' which we construct in two steps. First, we take a triangle-free circle graph H to be the graph whose circle representation is in the left part of Figure 7. The graph H is not 3-colorable, as shown by Gyárfás and Lehel [9]. We then remove the edge $\{m, n\}$ to create the graph H' that is still a circle graph and is 3-colorable, as shown in the right part of Figure 7. We observe that vertices m and n must have the same color in every 3-coloring of the graph H' – if there existed a 3-coloring c of H' so that $c(m) \neq c(n)$, c would also be a 3-coloring of the graph H, which is a contradiction.

As the graph H' has two vertices m, n which have the same color in every 3-coloring of H', we set all vertices that intersect the original L-shape's horizontal line segment to be the neighbors of the vertex m and similarly, we set all vertices intersecting the original L-shape's vertical line segment to be the neighbors of the vertex n. Since H' is a circle graph, it has an L-representation using only L-shapes with both of their endpoints on a circle [1]. Therefore we can arbitrarily extend the horizontal line segment of m and the vertical line segment of n as shown in Figure 8.

We easily observe that G is 3-colorable if and only if G' is. The reduction is obviously

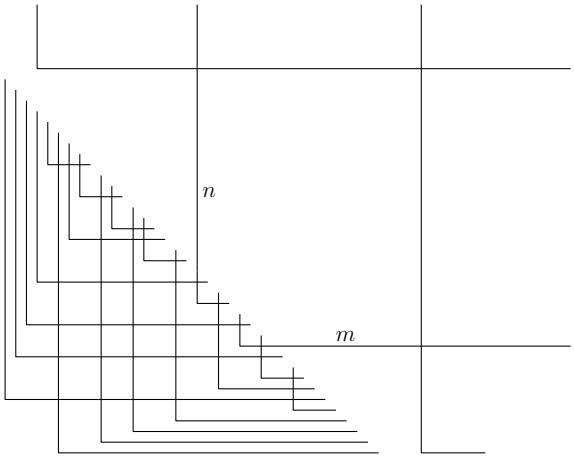


Figure 8 Replacing the vertex in the L-graph by the circle subgraph

polynomial, as every vertex is replaced by 18 vertices. ◀

References

1 Andrei Asinowski, Elad Cohen, Martin Charles Golumbic, Vincent Limouzy, Marina Lipshteyn, and Michal Stern. Vertex intersection graphs of paths on a grid. *Journal of Graph Algorithms and Applications*, 16(2):129–150, 2012.

2 Therese Biedl and Martin Derka. 1-String B_2-VPG Representation of Planar Graphs. In *31st International Symposium on Computational Geometry (SoCG 2015)*, volume 34 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 141–155. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2015.

3 Steven Chaplick, Vít Jelínek, Jan Kratochvíl, and Tomáš Vyskočil. Bend-bounded path intersection graphs: Sausages, noodles, and waffles on a grill. In *Graph-Theoretic Concepts in Computer Science*, pages 274–285. Springer, 2012.

4 Steven Chaplick and Torsten Ueckerdt. Planar graphs as VPG-graphs. In *Graph Drawing*, pages 174–186. Springer, 2013.

5 Stefan Felsner, Kolja Knauer, George B. Mertzios, and Torsten Ueckerdt. Intersection graphs of L-shapes and segments in the plane. *Discrete Applied Mathematics*, 206:48–55, 2016.

6 Michael R. Garey, David S. Johnson, and Larry J. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976.

7 Daniel Gonçalves, Lucas Isenmann, and Claire Pennarun. Planar graphs as L-intersection or L-contact graphs. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '18, pages 172–184. Society for Industrial and Applied Mathematics, 2018.

8 Herbert Grötzsch. Zur Theorie der diskreten Gebilde, VII: Ein Dreifarbensatz für dreikreisfreie Netze auf der Kugel. *Wiss. Z. Martin-Luther-Univ. Halle-Wittenberg Math.-Natur. Reihe*, 8:109–120, 1959.

9 András Gyárfás and Jenő Lehel. Covering and coloring problems for relatives of intervals. *Discrete Math.*, 55(2):167–180, 1985.

10 Jan Kratochvíl. A special planar satisfiability problem and a consequence of its NP-completeness. *Discrete Applied Mathematics*, 52(3):233–252, 1994.

Coloring Circle Arrangements: New 4-Chromatic Planar Graphs*

Man-Kwun Chiu[1], Stefan Felsner[2], Manfred Scheucher[2], Felix Schröder[2], Raphael Steiner[2], and Birgit Vogtenhuber[3]

1 **Institut für Informatik,**
Freie Universität Berlin, Germany
chiumk@zedat.fu-berlin.de
2 **Institut für Mathematik,**
Technische Universität Berlin, Germany
{felsner,scheucher,fschroed,steiner}@math.tu-berlin.de
3 **Institute of Software Technology,**
Graz University of Technology, Austria
bvogt@ist.tugraz.at

Abstract

Felsner, Hurtado, Noy and Streinu (2000) conjectured that arrangement graphs of simple great-circle arrangements have chromatic number at most 3. This paper is motivated by the conjecture.

We show that the conjecture holds in the special case when the arrangement is $\triangle$-*saturated*, i.e., arrangements where one color class of the 2-coloring of faces consists of triangles only. Moreover, we extend $\triangle$-saturated arrangements with certain properties to a family of arrangements which are 4-chromatic. The construction has similarities with Koester's (1985) crowning construction.

We also investigate fractional colorings. We show that every arrangement $\mathcal{A}$ of pairwise intersecting pseudocircles is "close" to being 3-colorable; more precisely $\chi_f(\mathcal{A}) \leq 3 + O(\frac{1}{n})$ where n is the number of pseudocircles. Furthermore, we construct an infinite family of 4-edge-critical 4-regular planar graphs which are fractionally 3-colorable. This disproves the conjecture of Gimbel, Kündgen, Li and Thomassen (2019) that every 4-chromatic planar graph has fractional chromatic number strictly greater than 3.

1 Introduction

An arrangement of pseudocircles is a family of simple closed curves on the sphere or in the plane such that each pair of curves intersects at most twice. Similarly, an arrangement of pseudolines is a family of x-monotone curves such that every pair of curves intersects exactly once. An arrangement is *simple* if no three pseudolines/pseudocircles intersect in a common point and *intersecting* if every pair of pseudolines/pseudocircles intersects. Given an arrangement of pseudolines/pseudocircles, the *arrangement graph* is the planar graph obtained by placing vertices at the intersection points of the arrangement and thereby subdividing the pseudolines/pseudocircles into edges.

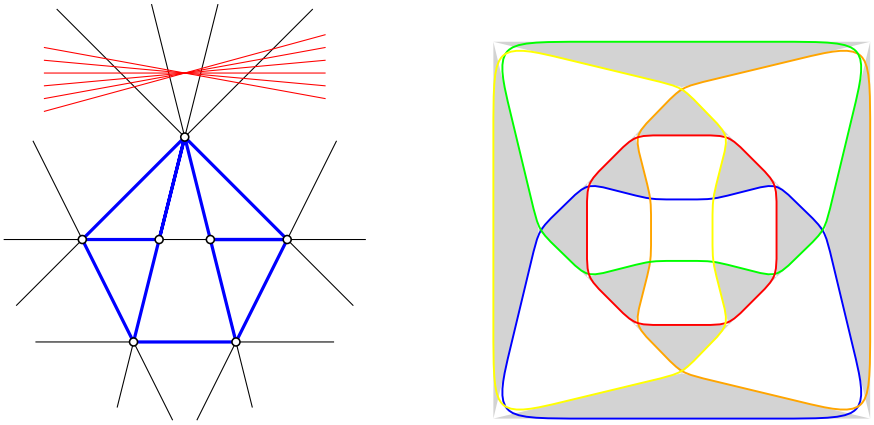
A *(proper) coloring* of a graph assigns a color to each vertex such that no two adjacent vertices have the same color. The *chromatic number* χ is the smallest number of colors

* M.-K. Chiu was supported by ERC StG 757609. S. Felsner was supported by DFG Grant FE 340/13-1. M. Scheucher was supported by the internal research funding "Post-Doc-Funding" from Technische Universität Berlin. R. Steiner was supported by DFG-GRK 2434. B. Vogtenhuber was supported by the FWF project I 3340-N35. This work was initiated at a workshop of the collaborative DACH project *Arrangements and Drawings* in Malchow, Mecklenburg-Vorpommern. We thank the organizers and all the participants for the inspiring atmosphere.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

needed for a proper coloring. The famous 4-color theorem and also Brook's theorem imply the 4-colorability of planar graphs with maximum degree 4. This motivates the question: which arrangement graphs need 4 colors in any proper coloring?

There exist arbitrarily large non-simple line arrangements that require 4 colors. For example, the construction depicted in Figure 1(a) contains the Moser spindle as subgraph and hence cannot be 3-colored. Using an inverse central (gnomonic) projection, which maps lines to great-circles, one gets non-simple arrangements of great-circles with $\chi = 4$. Koester [12] presented a simple arrangement of 7 circles with $\chi = 4$ in which all but one pair of circles intersect, see Figure 3(b). Moreover, there are simple intersecting arrangements that require 4 colors. We invite the reader to verify this property for the example depicted in Figure 1(b).



(a) (b)

Figure 1 (a) A 4-chromatic non-simple line arrangement. The red subarrangement not intersecting the Moser spindle (highlighted blue) can be chosen arbitrarly. (b) A simple intersecting arrangement of 5 pseudocircles with $\chi = 4$ and $\chi_f = 3$.

In 2000, Felsner, Hurtado, Noy and Streinu [3] (cf. [4]) studied arrangement graphs of pseudoline and pseudocircle arrangements. They have results regarding connectivity, Hamiltonicity, and colorability of those graphs. In this work, they also stated the following conjecture:

▶ **Conjecture 1** (Felsner et al. [3, 4]). *The arrangement graph of every simple arrangement of great-circles is 3-colorable.*

While the conjecture is fairly well known (cf. [15, 10, 19] and [20, Chapter 17.7]) there has been little progress in the last 20 years. Aichholzer, Aurenhammer, and Krasser verified the conjecture for up to 11 great-circles [13, Chapter 4.6.4].

Results and outline

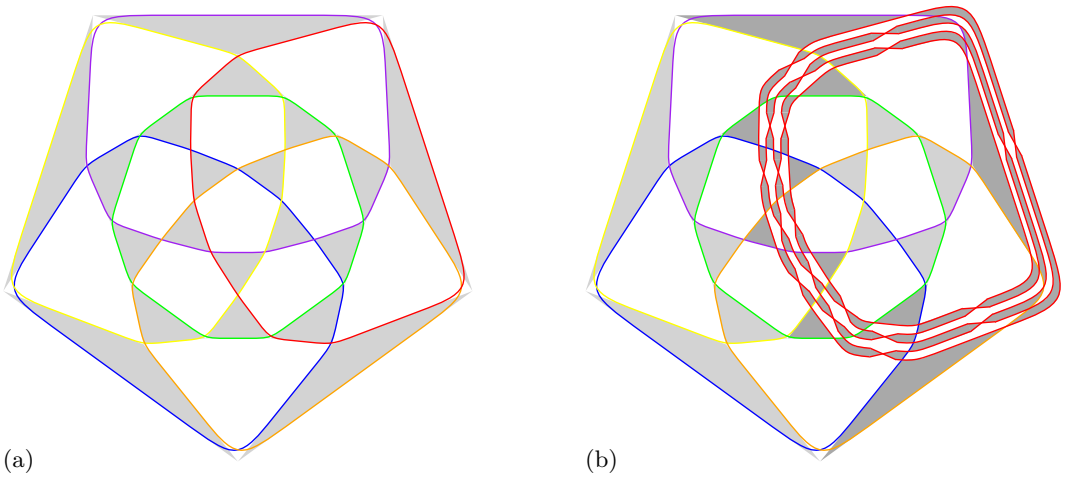
In Section 2 we show that Conjecture 1 holds for $\triangle$-*saturated* arrangements of pseudocircles, i.e., arrangements where one color class of the 2-coloring of faces consists of triangles only. In Section 3 we extend our study of $\triangle$-saturated arrangements and present an infinite family of arrangements which require 4 colors. The construction generalizes Koester's [12] arrangement of 7 circles which requires 4 colors; see Figure 3(b). Moreover, we believe that the

construction results in infinitely many 4-vertex-critical[1] arrangement graphs. Koester [12] obtained his example using a "crowning" operation, which actually yields infinite families of 4-edge-critical 4-regular planar graphs. However, except for the 7 circles example these graphs are not arrangement graphs.

In Section 4 we investigate the fractional chromatic number χ_f of arrangement graphs. This variant of the chromatic number is the objective value of the linear relaxation of the ILP formulation for the chromatic number. We show that intersecting arrangements of pseudocircles are "close" to being 3-colorable by proving that $\chi_f(\mathcal{A}) \leq 3 + O(\frac{1}{n})$ where n is the number of pseudocircles of $\mathcal{A}$. In Section 5, we present an example of a 4-edge-critical arrangement graph which is fractionally 3-colorable. The example is the basis for constructing an infinite family of 4-regular planar graphs which are 4-edge-critical and fractionally 3-colorable. This disproves Conjecture 3.2 from Gimbel, Kündgen, Li and Thomassen [7] that every 4-chromatic planar graph has fractional chromatic number strictly greater than 3. In Section 6 we report on our computational data, mention some some new observations related to Conjecture 1, and present strengthened versions of the conjecture.

2 △-saturated arrangements are 3-colorable

The maximum number of triangles in arrangements of pseudolines and pseudocircles has been studied intensively, see e.g. [8, 16, 2] and [6]. By recursively applying the "doubling method", Harborth [9] and also [16, 2] proved the existence of infinite families of △-saturated arrangements of pseudolines. Similarly, a doubling construction for arrangements of (great-)pseudocircles yields infinitely many △-saturated arrangements of (great-) pseudocircles. Figure 2 illustrates the doubling method applied to an arrangement of great-pseudocircles. It will be relevant later that arrangements obtained via doubling contain pentagonal cells. Note that for $n \equiv 2 \pmod 3$ there is no △-saturated intersecting pseudocircle arrangement because the number of edges of the arrangement graph is not divisible by 3.



(a) (b)

Figure 2 The doubling method applied to an arrangements of 6 great-pseudocircle. The red pseudocircle is replaced by a cyclic arrangement. Triangular cells are shaded gray.

[1] A k-chromatic graph is *k-vertex-critical* if the removal of every vertex decreases the chromatic number. It is *k-edge-critical* if the removal of every edge decreases the chromatic number.

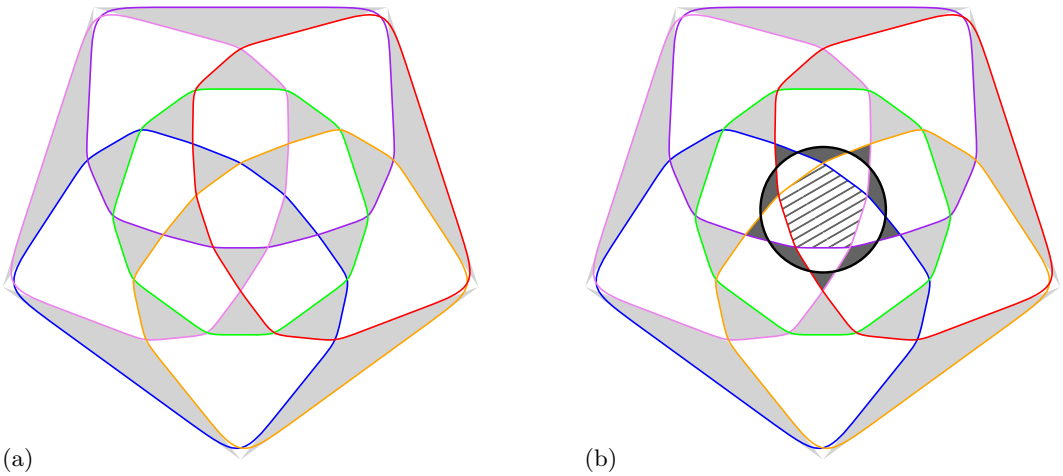
▶ **Theorem 2.** *Every △-saturated arrangement $\mathcal{A}$ of pseudocircles is 3-colorable.*

Proof. Let H be a graph whose vertices correspond to the triangles of $\mathcal{A}$ and whose edges correspond to pairs of triangles sharing a vertex of $\mathcal{A}$. This graph H is planar and 3-regular. Moreover, since the arrangement graph of $\mathcal{A}$ is 2-connected, H is bridgeless. Now Tait's theorem, a well known equivalent of the 4-color theorem, asserts that H is 3-edge-colorable, see e.g. [1] or [18]. The edges of H correspond bijectively to the vertices of the arrangement $\mathcal{A}$ and, since adjacent vertices of $\mathcal{A}$ are incident to a common triangle, the corresponding edges of H share a vertex. This shows that the graph of $\mathcal{A}$ is 3-colorable. ◀

3 Constructing 4-chromatic arrangement graphs

In this section, we describe an operation that extends any △-saturated intersecting arrangement of pseudocircles with a pentagonal cell (which is 3-colorable by Theorem 2) to a 4-chromatic arrangement of pseudocircles by inserting one additional pseudocircle.

The corona extension: We start with a △-saturated arrangement of pseudocircles which contains a pentagonal cell ⬠. By definition, in the 2-coloring of the faces one of the two color classes consists of triangles only; see e.g. the arrangement from Figure 3(a). Since the arrangement is △-saturated, the pentagonal cell ⬠ is surrounded by triangular cells. As illustrated in Figure 3(b) we can now insert an additional pseudocircle close to ⬠. This newly inserted pseudocircle intersects only the 5 pseudocircles which bound ⬠, and in the so-obtained arrangement one of the two dual color classes consists of triangles plus the pentagon ⬠. It is interesting to note that the arrangement depicted in Figure 3(b) is precisely Koester's arrangement [11, 12].



(a) (b)

Figure 3 (a) A △-saturated arrangement of 6 great-circles and (b) the corona extension at its central pentagonal face. The arrangement in (b) is Koester's [11] example of a 4-edge-critical 4-regular planar graph.

The following proposition plays a central role in this section.

▶ **Proposition 3.** *The corona extension of a △-saturated arrangement of pseudocircles with a pentagonal cell ⬠ is 4-chromatic.*

The proof is based on the observation that after the corona extension the inequality $3\alpha < |V|$ holds.

By applying the corona extension to members of the infinite family of $\triangle$-saturated arrangements with pentagonal cells (cf. Section 2), we obtain an infinite family of arrangements that are not 3-colorable.

▶ **Theorem 4.** *There exists an infinite family of 4-chromatic arrangements of pseudocircles.*

Koester [12] defines a related construction which he calls *crowning* and constructs his example by two-fold crowning of a graph on 10 vertices. He also uses crowning to generate an infinite family of 4-edge-critical 4-regular graphs. In the full version of our paper, we present sufficient conditions to obtain a 4-vertex-critical arrangement via the corona extension. We conclude this section with the following conjecture:

▶ **Conjecture 5.** *There exists an infinite family of arrangement graphs of arrangements of pseudocircles that are 4-vertex-critical.*

4 Fractional colorings

In this section, we investigate fractional colorings of arrangements. A *b-fold coloring* of a graph G with m colors is an assignment of a set of b colors from $\{1, \ldots, m\}$ to each vertex of G such that the color sets of any two adjacent vertices are disjoint. The *b-fold chromatic number* $\chi_b(G)$ is the minimum m such that G admits a b-fold coloring with m colors. The *fractional chromatic number* of G is $\chi_f(G) := \lim_{b\to\infty} \frac{\chi_b(G)}{b} = \inf_b \frac{\chi_b(G)}{b}$. With α being the independence number and ω being the clique number, the following inequalities hold:

$$\max\left\{\frac{|V|}{\alpha(G)}, \omega(G)\right\} \leq \chi_f(G) \leq \frac{\chi_b(G)}{b} \leq \chi(G). \tag{1}$$

▶ **Theorem 6.** *Let G be the arrangement graph of an intersecting arrangement $\mathcal{A}$ of n pseudocircles, then $\chi_f(G) \leq 3 + \frac{6}{n-2}$.*

Sketch of the proof. Let C be a pseudocircle of $\mathcal{A}$. After removing all vertices along C from the arrangement graph G we obtain a graph which has two connected components A (vertices in the interior of C) and B (vertices in the exterior). Let C' be a small circle contained in one of the faces of A, the Sweeping Lemma of Snoeyink and Hershberger [17] asserts that there is a continuous transformation of C' into C which traverses each vertex of A precisely once. In particular, when a vertex is traversed, at most two of its neighbors have been traversed before. Hence, we obtain a 3-coloring of the vertices of A by greedily coloring vertices in the order in which they occur during the sweep. An analogous argument applies to B. Taking such a partial 3-coloring of G for each of the n pseudocircles of $\mathcal{A}$, we obtain for each vertex a set of $n-2$ colors, i.e., an $(n-2)$-fold coloring of G. The total number of colors used is $3n$. The statement now follows from inequality (1). ◀

5 Fractionally 3-colorable 4-edge-critical planar graphs

From our computational data (cf. [5]), we observed that some of the arrangements such as the 20 vertex graph depicted in Figure 1(b) have $\chi = 4$ and $\chi_f = 3$, and therefore disprove Conjecture 3.2 by Gimbel et al. [7][2]. Moreover, we determined that there are precisely

[2] Computing the fractional chromatic number of a graph is NP-hard in general [14]. For our computations we formulated a linear program which we then solved using the MIP solver Gurobi.

17 4-regular 18-vertex planar graphs with $\chi = 4$ and $\chi_f = 3$, which are minimal in the sense that there are no 4-regular graphs on $n \leq 17$ vertices with $\chi = 4$ and $\chi_f = 3$. Each of these 17 graphs is 4-vertex-critical and the one depicted in Figure 4(a) is even 4-edge-critical.

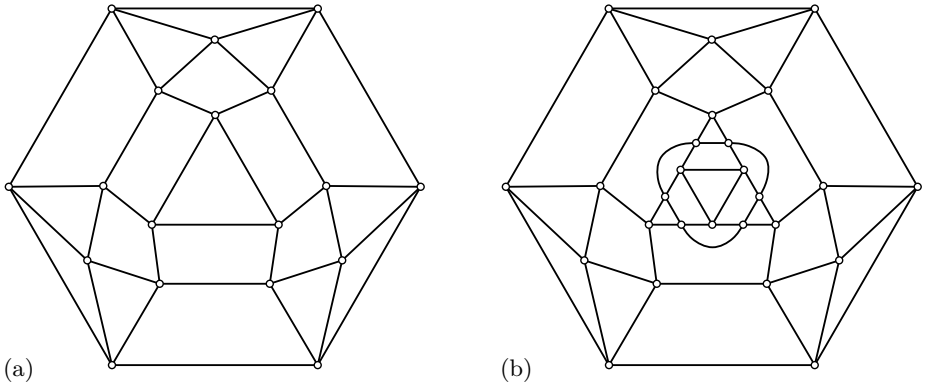


Figure 4 (a) A 4-edge-critical 4-regular 18-vertex planar graph with $\chi = 4$ and $\chi_f = 3$ and (b) the crowning extension at its center triangular face.

Starting with a triangular face in the 4-edge-critical 4-regular graph depicted in Figure 4(a) and repeatedly applying the Koester's crowning operation [12] as illustrated in Figure 4(b) (which by definition preserves the existence of a facial triangle), we deduce the following theorem.

▶ **Theorem 7.** *There exists an infinite family of 4-edge-critical 4-regular planar graphs G with fractional chromatic number $\chi_f(G) = 3$.*

6 Discussion

With Theorem 2 we gave a proof of Conjecture 1 for $\triangle$-saturated great-pseudocircle arrangements. While this is a very small subclass of great-pseudocircle arrangements, it is reasonable to think of it as a "hard" class for 3-coloring. The rationale for such thoughts is that triangles restrict the freedom of extending partial colorings. Our computational data indicates that sufficiently large intersecting pseudocircle arrangements that are *diamond-free*, i.e., no two triangles of the arrangement share an edge, are also 3-colorable. Computations also suggest that sufficiently large great-pseudocircle arrangements have *antipodal colorings*, i.e., 3-colorings where antipodal points have the same color. Based on the experimental data we propose the following strengthened variants of Conjecture 1.

▶ **Conjecture 8.** *The following three statements hold.*

(a) Every diamond-free intersecting arrangement of $n \geq 6$ pseudocircles is 3-colorable.
(b) Every intersecting arrangement of sufficiently many pseudocircles is 3-colorable.
(c) Every arrangement of $n \geq 7$ great-pseudocircles has an antipodal 3-coloring.

References

1 M. Aigner. *Graph theory. A development from the 4-color problem.* BCS Assoc, 1987.
2 J. Blanc. The best polynomial bounds for the number of triangles in a simple arrangement of n pseudo-lines. *Geombinatorics*, 21:5–17, 2011.

3 S. Felsner, F. Hurtado, M. Noy, and I. Streinu. Hamiltonicity and colorings of arrangement graphs. In *Proceedings of the* 11^{th} *annual ACM-SIAM symposium on Discrete algorithms (SODA'00)*, pages 155–164, 2000.

4 S. Felsner, F. Hurtado, M. Noy, and I. Streinu. Hamiltonicity and colorings of arrangement graphs. *Discrete Applied Mathematics*, 154(17):2470–2483, 2006.

5 S. Felsner and M. Scheucher. Webpage: Homepage of pseudocircles. `http://www3.math.tu-berlin.de/pseudocircles`.

6 S. Felsner and M. Scheucher. Arrangements of Pseudocircles: Triangles and Drawings. *Discrete & Computational Geometry*, 65:261–278, 2021.

7 J. Gimbel, A. Kündgen, B. Li, and C. Thomassen. Fractional coloring methods with applications to degenerate graphs and graphs on surfaces. *SIAM Journal on Discrete Mathematics*, 33(3):1415–1430, 2019.

8 B. Grünbaum. *Arrangements and Spreads*, volume 10 of *CBMS Regional Conference Series in Mathematics*. American Mathematical Society, 1972 (reprinted 1980).

9 H. Harborth. Some simple arrangements of pseudolines with a maximum number of triangles. *Annals of the New York Academy of Sciences*, 440(1):31–33, 1985.

10 G. Kalai. Coloring problems for arrangements of circles (and pseudocircles): Eight problems on coloring circles, 2018. `http://gilkalai.wordpress.com/2018/04/13/coloring-problems-for-arrangements-of-circles-and-pseudocircles/`.

11 G. Koester. Note to a problem of T. Gallai and G. A. Dirac. *Combinatorica*, 5:227–228, 1985.

12 G. Koester. 4-critical 4-valent planar graphs constructed with crowns. *Math. Scand.*, 67:15–22, 1990.

13 H. Krasser. *Order Types of Point Sets in the Plane*. PhD thesis, Institute for Theoretical Computer Science, Graz University of Technology, Austria, 2003.

14 C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM*, 41(5):960–981, 1994.

15 Open Problem Garden. 3-colourability of arrangements of great circles, 2009. `http://www.openproblemgarden.org/op/3_colourability_of_arrangements_of_great_circles`.

16 J.-P. Roudneff. On the number of triangles in simple arrangements of pseudolines in the real projective plane. *Discrete Mathematics*, 60:243–251, 1986.

17 J. Snoeyink and J. Hershberger. Sweeping arrangements of curves. In *Discrete & Computational Geometry: Papers from the DIMACS Special Year*, volume 6 of *Series in Discrete Mathematics and Theoretical Computer Science*, pages 309–349. American Mathematical Society, 1991.

18 R. Thomas. An update on the four-color theorem. *Notices of the American Mathematical Society*, 45:848–859, 1998.

19 S. Wagon. A machine resolution of a four-color hoax. In *Abstracts for the* 14^{th} *Canadian Conference on Computational Geometry (CCCG'02)*, pages 181–192, 2002.

20 S. Wagon. *Mathematica in Action: Problem Solving Through Visualization and Computation*. Springer, 2010.

Minimum-Error Triangulation is NP-hard

Anna Arutyunova[*1], Anne Driemel[†2], Jan-Henrik Haunert[3], Herman Haverkort[4], Petra Mutzel[†5], and Heiko Röglin[*†6]

1 **Institute of Computer Science, University of Bonn, Germany**
arutyunova@informatik.uni-bonn.de
2 **Hausdorff Center for Mathematics, University of Bonn, Germany**
driemel@cs.uni-bonn.de
3 **Institute of Geodesy and Geoinformation, University of Bonn, Germany**
haunert@igg.uni-bonn.de
4 **Institute of Computer Science, University of Bonn, Germany**
cs.herman@haverkort.net
5 **Institute of Computer Science, University of Bonn, Germany**
petra.mutzel@cs.uni-bonn.de
6 **Institute of Computer Science, University of Bonn, Germany**
roeglin@cs.uni-bonn.de

Abstract

Given two sets of points in the plane, P and R, with a real-valued function defined on $P \cup R$, we define for any triangulation defined on P the vertical error at the points of R and wish to find a triangulation that minimizes this error. We show that this problem is NP-hard to approximate within any approximation factor.

1 Introduction

Let $P \subset \mathbb{R}^2$ be a set of n points and $f\colon P \to \mathbb{R}$. We call P the set of *triangulation points* and $f(p)$ the *measurement value* of $p \in P$. Additionally we are given a set $R \subset \text{conv}(P)$ of m points and a function $h\colon R \to \mathbb{R}$. We refer to R as the set of *reference points* and to $h(r)$ as the *reference value* of $r \in R$.
A triangulation of P is a maximal set of non-crossing edges between points in P. Such a triangulation D defines an extension of f on points in $\text{conv}(P)$ by linearly interpolating f in every triangle. This way we obtain a piece-wise linear function $s_D\colon \text{conv}(P) \to \mathbb{R}$.

The **min-error triangulation problem** asks for a triangulation D of P such that the L_p-error $(\text{Err}_D(R))^{\frac{1}{p}}$ between the reference values and the interpolation is minimized, where

$$\text{Err}_D(M) = \sum_{r \in M} |s_D(r) - h(r)|^p$$

is the error of D on $M \subseteq R$. The **zero-error triangulation problem** asks whether there is a triangulation with zero L_p-error. We prove that this problem is NP-hard.

▶ **Theorem 1.** *The zero-error triangulation problem is NP-hard for any value of $p \geq 1$. Thus the min-error triangulation problem cannot be approximated within any multiplicative factor in polynomial time unless $P = NP$.*

* Supported by DFG grant RO 5439/1-1
† Supported by the Hausdorff Center for Mathematics at the University of Bonn (DFG GZ 2047/1, project ID 390685813)

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

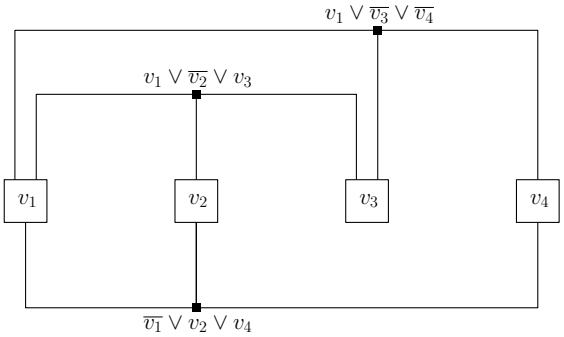


Figure 1 Embedding of the 3SAT formula $(\overline{v_1} \vee v_2 \vee v_4) \wedge (v_1 \vee \overline{v_2} \vee v_3) \wedge (v_1 \vee \overline{v_3} \vee \overline{v_4})$.

2 Related Work

Triangulating sets of points in the plane is a fundamental task of computational geometry. It is of high relevance for data interpolation and surface modelling tasks, where for every data point a data value (or height) is given in addition to the point's two plane coordinates x and y. The Delaunay triangulation is most often applied as it optimizes several relevant criteria and can be computed efficiently. In particular, it maximizes the minimum angle among all the angles of all the triangles, measured in the xy-plane [7]. In the context of surface modelling, one of the most interesting properties of the Delaunay triangulation is that it minimizes the Dirichlet energy (and thus the mean squared slope) of the piece-wise linear surface defined from the data points, their data values as heights, and the triangulation [11]. This result is surprising, since computing the Delaunay triangulation does not require knowledge of the data values. *Data dependent triangulations*, on the other hand, have been defined in [4] as triangulations that are computed under consideration of the data values. As optimization criteria the authors have considered (1) smoothness criteria such as minimizing the jump in normal derivatives in the edges of the triangulation, (2) criteria based on three-dimensional properties of the triangles such as maximizing the minimum angle of the triangles in the three-dimensional space, (3) variational criteria based on a function generalizing the Dirichlet energy, and (4) min-error criteria measuring the quality of the triangulation by comparison with a reference surface or reference point set. Data dependent triangulations of the latter type, in the following referred to as min-error triangulations, are the subject of this paper.

There are many heuristics for computing data-dependent triangulations [2, 3, 4, 12], which are usually based on Lawson's edge flip algorithm [7]. The problem can be solved based on integer linear programming [10]. Using established techniques, exact polynomial-time algorithms can be obtained for some special or restricted cases [5]. However, prior to our work, little was known about the complexity of computing or approximating min-error triangulations in the general case. For related problems some hardness results exist [1, 9].

3 Reduction

We prove that the zero-error triangulation problem is NP-hard by a reduction from the planar 3SAT problem, which was proven to be NP-complete by Lichtenstein [8]. An instance of this problem can be embedded into the plane, where every clause is represented by a vertex and every variable by a box placed on the horizontal axis. A box is connected to a vertex via a rectilinear edge if the respective variable is contained in the clause. For an example, see Figure 1. Such an embedding is for example also used in [6].

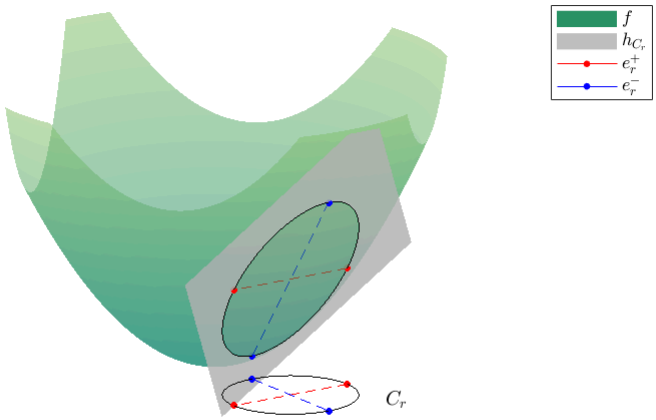


Figure 2 Example of a reference point r with coupled circle C_r and its positive/negative edges crossing at r. Lifting the red and blue points to $\mathbb{R}^3$, with their measurement values as third coordinate, we see that these points lie on both the paraboloid and the plane containing $(r, h_{C_r}(r))$.

For every instance of the planar 3SAT problem we construct an instance for the zero-error triangulation problem by replacing the boxes, vertices and edges of its rectilinear embedding in the plane by a set of triangulation points and reference points. For this purpose we handle each component of the 3SAT embedding individually. We construct the *variable gadgets* which replace the boxes, the *wire gadgets*, which replace the rectilinear edges and finally the *clause gadgets* and the *negation gadgets*, where the first replace the vertices and the second can be attached to variable gadgets to handle negated variables in a clause. The combination of these gadgets then constitute an instance to the zero-error triangulation problem.

We ensure that there are two possible zero-error triangulations on the reference points belonging to a variable gadget and the attached negation gadgets and wire gadgets as follows. Points from P together with their measurement value can be seen as points in $\mathbb{R}^3$. We ensure that they lie on a paraboloid in $\mathbb{R}^3$ and exploit the properties of the paraboloid (its convexity and the correspondence of planes in $\mathbb{R}^3$ to circles in $\mathbb{R}^2$) to limit possible zero-error triangulations. Any such triangulation then corresponds to the assignment of value 0 (negative) or 1 (positive) to any variable. We claim that the instance can be triangulated with zero error if and only if the 3SAT instance is solvable. The usage of the paraboloid differentiates our reduction from previous work in this area.

4 Preliminaries

Our triangulation instance consists of a set of triangulation points with integral coordintes $P \subset \mathbb{Z}^2$ and a set $R \subset \text{conv}(P)$ of reference points. The measurement value of a triangulation point $p = (p_1, p_2) \in P$ is given by $f(p) = p_1^2 + p_2^2$. On the other hand reference values are not determined by one single function. Instead we define a set of functions, one for every circle in $\mathbb{R}^2$, and choose for every reference point one of these functions which determines the reference value of this point. Concretely let C be a circle around a point $x = (x_1, x_2)$ with radius ρ. We denote with $I_C = \{y \in \mathbb{R}^2 \mid \|x - y\| < \rho\}$ the *interior* of C and with $O_C = \mathbb{R}^2 \backslash (C \cup I_C)$ the *exterior* of C. For a reference point $r = (r_1, r_2) \in R$ we define the function

$$h_C(r) = 2x_1r_1 + 2x_2r_2 - x_1^2 - x_2^2 + \rho^2.$$

The function graph of f is the unit paraboloid $\{(p_1, p_2, p_1^2 + p_2^2) \mid (p_1, p_2) \in \mathbb{R}^2\}$ and the function graph of h_C is the plane containing the lifting of C onto the paraboloid (Figure 2).

Every point $r \in R$ is then *coupled* to a circle, which we denote by C_r. It will be defined during the construction of the gadgets and determines the reference value $h(r) = h_{C_r}(r)$. Let an edge denote a set of two points in $\mathbb{R}^2$. For each r we define a *positive edge* e_r^+ and a *negative edge* e_r^- both consisting of triangulation points lying on C_r and intersecting each other at r (i.e., $\text{conv}(e_r^+) \cap \text{conv}(e_r^-) = \{r\}$). Figure 2 shows the whole construction. We say for a triangulation D that the *signal* at $r \in R$ is *positive* if D contains edge e_r^+ and *negative* if it contains e_r^-, otherwise we call it *ambiguous*. Similarly we call D *positive* on $M \subset R$ if the signal at all $r \in M$ is positive and *negative* if the signal at all $r \in M$ is negative.

We interpret a triangle as the set of its vertices and say that a triangle $T = \{s, t, u\} \subset P$ is in D if all of $\{s, t\}, \{t, u\}, \{u, s\}$ are in D and $\text{conv}(T)$ does not contain further triangulation points, i.e., $\text{conv}(T) \cap P = T$. We say that $r \in R$ is *represented with zero error* by T if $r \in \text{conv}(T)$ and the value at r of the linear interpolation of f on $\text{conv}(T)$ equals $h(r)$.

▶ **Lemma 2.** *Let r be a point of R and let $T \subset \mathbb{R}^2$ be a triangle with $r \in \text{conv}(T \cap C_r)$. Then r is represented with zero error by T.*

If the 3SAT instance is satisfiable, we argue that there is a triangulation containing one of $e_r^\pm$ for every reference point r. Lemma 2 states that such a triangulation has in fact zero error (see also Figure 2). To represent r with zero error in any other way, we need at least one triangulation point inside and one outside C_r. This follows from the convexity of f.

▶ **Lemma 3.** *Let $T \subset \mathbb{R}^2$ be a triangle representing $r \in R$ with zero error. If $r \notin \text{conv}(T \cap C_r)$, then T has a non-empty intersection with I_{C_r} and O_{C_r}.*

This ensures that every zero-error triangulation yields a solution to the 3SAT instance. In particular, we guarantee during the construction that only few triangulation points lie inside C_r for each reference point r. With a concise case analysis we rule out that any of them can be used together with a point outside C_r to form a triangle that represents r with zero error, which limits the choice to triangles containing one of $e_r^\pm$.

Finally, our instance contains a set of *mandatory edges* that must be part of any feasible triangulation of P. Mandatory edges are not part of the zero-error triangulation problem, but they can be replaced afterwards by an additional construction.

5 The Gadgets

At the core of our reduction lies the design of the gadgets which constitute the triangulation instance. Before we dedicate ourselves to the more complicated gadgets we construct smaller elements called *bits* and *segments* which then are combined into the larger gadgets.

A *bit* occupies a small construction around a central point in $\mathbb{Z}^2$ and can be oriented either horizontally or vertically. We describe the horizontal bit at a reference point $r \in R$, which is also the only reference point of this bit. It is coupled to a circle C_r which is centered on r and has radius $\sqrt{2}$. The integer grid points on this circle, that is, the points $r + (\pm 1, \pm 1)$, are triangulation points. Moreover $r + (0, 1)$ and $r + (0, -1)$ are triangulation points, whereas $r + (-2, 0), r + (-1, 0), r + (1, 0)$ and $r + (2, 0)$ are *not*. Therefore, we call the latter points *forbidden*. Furthermore we define the positive and negative edge as

$$e_r^+ = \{r + (-1, -1), r + (1, 1)\}, \quad e_r^- = \{r + (-1, 1), r + (1, -1)\}.$$

As $r + (\pm 1, \pm 1) \in C_r$, any triangle containing either e_r^+ or e_r^- represents r with zero error by Lemma 2. The vertical bit is defined similarly. Figure 3 illustrates both constructions.

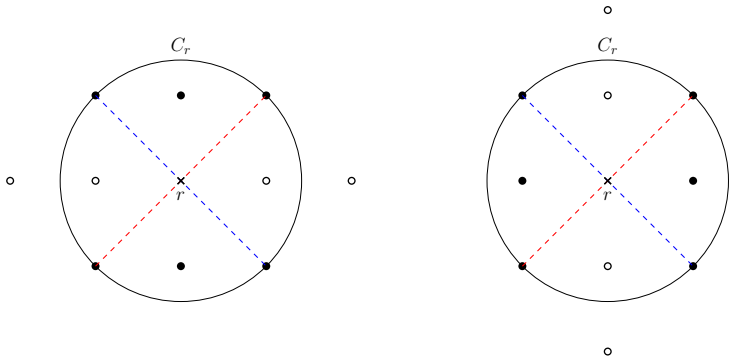


Figure 3 The (horizontal/vertical) bit at r with the positive edge in red and the negative edge in blue. The black points are triangulation points and the white points are forbidden.

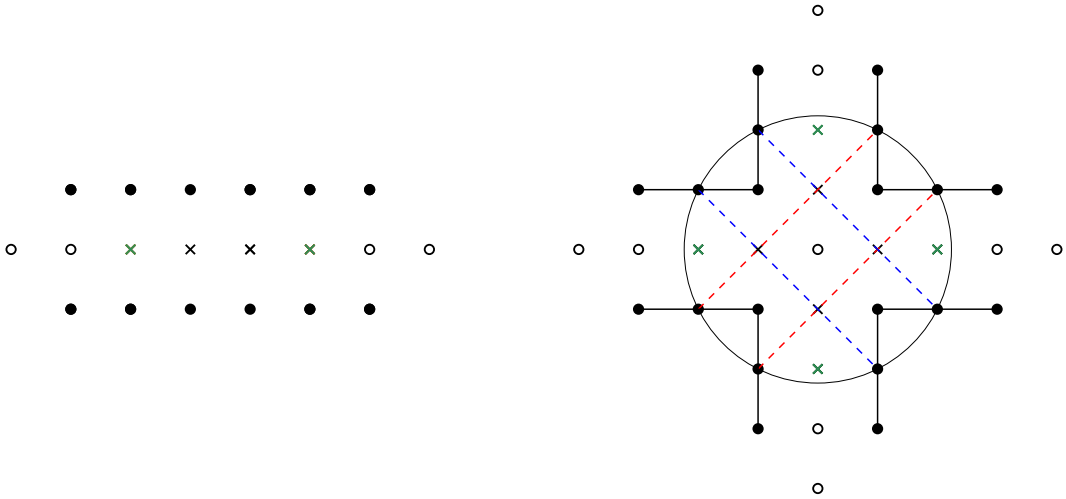


Figure 4 Example of a horizontal wire segment on the left and a multiplier segment with mandatory edges on the right. The red or blue edges indicate the positive or negative edges of the crossing points, respectively. All white points and all reference points are forbidden. The green points are anchor points.

▶ **Lemma 4.** *Suppose the instance contains a bit at r. If $P \subset \mathbb{Z}^2$ and P does not contain forbidden points of the bit, any triangulation D of P with $\mathrm{Err}_D(r) = 0$ contains one of $e_r^{\pm}$.*

The next larger components are the *wire segment* and the *multiplier segment*, which we build from bits. They can be combined at specified reference points, which we call *anchor points*. These points are always reference points of bits.

A *wire segment* connects two points $x, y \in \mathbb{Z}^2$ lying on the same horizontal or vertical line. We place a horizontal or vertical bit on x, y and all integral points lying between these on the line connecting x and y. The anchor points of this segment are x, y.

A *multiplier segment* at a point $x \in \mathbb{Z}^2$ consist of two horizontal bits at $x \pm (2, 0)$ and two vertical bits at $x \pm (0, 2)$. These four points are simultaneously anchor points. Furthermore we add four inner reference points $x \pm (0, 1), x \pm (1, 0)$ whose coupled circle is of radius $\sqrt{5}$ and centered around x. Figure 4 shows the wire segment and the multiplier segment including mandatory edges and the positive/negative edges of the inner reference points.

To obtain the larger variable gadget and wire gadget we combine wire segments with

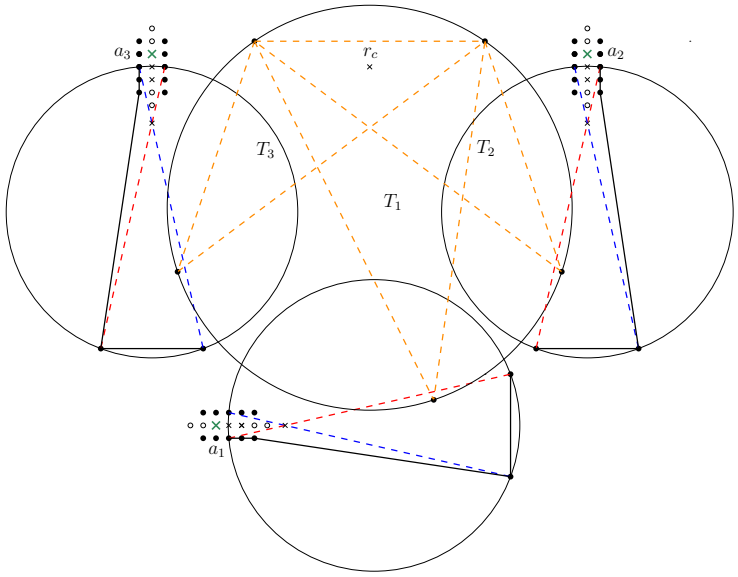


Figure 5 The clause gadget, where the red/blue edges indicate the positive/negative edges of the crossing points. The triangles T_1, T_2, T_3 are orange and the anchor points a_1, a_2, a_3 green.

multiplier segments. Two segments can be combined if they share a common anchor point. By the combination of two segments we mean the union of their reference points and triangulation points. A point is forbidden in the combination if it is forbidden in at least one of the segments. Thus it is not allowed to combine two segments if a triangulation point of one is forbidden in the other. The set of anchor points of the combination is defined as the symmetric difference of anchor point sets of both segments. This way we can combine arbitrarily many segments.

Remember that the *wire gadget* replaces the rectilinear edges of the 3SAT embedding, so it has to connect two points $x = (x_1, x_2), y = (y_1, y_2) \in \mathbb{Z}^2$. It consists of a multiplier segment placed on either (x_1, y_2) or (y_1, x_2), which is connected on two of its anchor points via two wire segments to both x and y. A *variable gadget* at $v \in \mathbb{Z}^2$ consists of m multiplier segments at sufficiently large distance $\alpha \in \mathbb{Z}$, which we do not specify further. Here m denotes the number of clauses. Concretely, we place a multiplier segment on each of the points $v + (k\alpha, 0)$ with $0 \leq k \leq m - 1$ and connect them via horizontal wire segments at their anchor points. The multiplier segments ensure that the gadget can later be connected at its anchor points to multiple clause gadgets. We observe that the described combinations of segments for both gadgets are allowed and that they have the following crucial property.

▶ **Lemma 5.** *Suppose the instance contains a wire/variable gadget and let $\widetilde{R}$ be the reference points of this gadget. If $P \subset \mathbb{Z}^2$ and P does not contain forbidden points of the gadget, any triangulation D of P with $\mathrm{Err}_D(\widetilde{R}) = 0$ is either positive or negative on $\widetilde{R}$.*

The *clause gadget* at a point $c \in \mathbb{Z}^2$ must combine three signals. To this end we add a reference point r_c. Instead of a positive/negative edge it comes with three triangles T_1, T_2, T_3 lying on C_{r_c}, each triangulating r_c with zero error. The clause gadget can be connected to other gadgets at three anchor points a_1, a_2, a_3. We add an additional construction that blocks the triangle T_i if the signal at a_i is positive for $i = 1, 2$ and T_3 if the signal at a_3 is negative. For the whole clause gadget we refer to Figure 5.

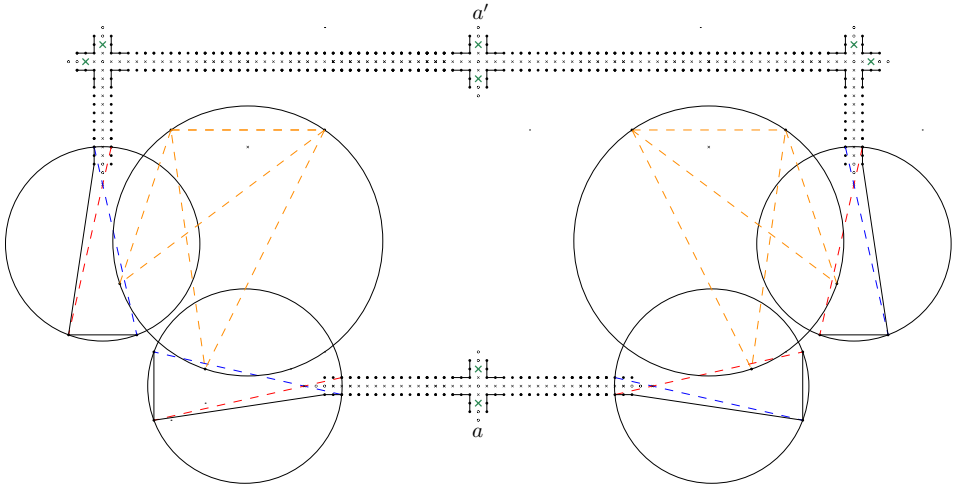


Figure 6 The negation gadget. If the signal at anchor point a is negative it is negated in the left segment. If the signal at a is positive it is negated in the right segment. Since the top wire carries a consistent signal, negation is ensured at a'.

▶ **Lemma 6.** *Suppose the instance contains a clause gadget and let $\widetilde{R}$ be its reference points. If $P \subset \mathbb{Z}^2$ and P does not contain forbidden points of the gadget, any triangulation D of P with $\mathrm{Err}_D(\widetilde{R}) = 0$ must be negative on one of the anchor points a_1, a_2 or positive on a_3.*

The last part of our construction is the *negation gadget*. It consist of some wire gadgets and two *functional segments*. Both functional segments are variations of the clause gadget, which combine two instead of three signals. The whole negation gadget is depicted in Figure 6. The next lemma shows that the signal at a' is indeed the negation of the signal at a.

▶ **Lemma 7.** *Suppose the instance contains a negation gadget and let $\widetilde{R}$ be the reference points of this gadget. Let $P \subset \mathbb{Z}^2$ and assume P does not contain forbidden points of the gadget. Any triangulation D of P with $\mathrm{Err}_D(\widetilde{R}) = 0$ is positive at a iff it is negative at a'.*

6 Reduction of Planar 3SAT

▶ **Theorem 1.** *The zero-error triangulation problem is NP-hard for any value of $p \geq 1$. Thus the min-error triangulation problem cannot be approximated within any multiplicative factor in polynomial time unless $P = NP$.*

For an instance of the planar 3SAT problem we construct the corresponding zero-error triangulation instance from the gadgets presented above. With an additional construction we are able to delete all mandatory edges from the instance and argue that Lemmas 5-7 still hold after the deletion. Then Lemmas 5-7 guarantee that the 3SAT formula is satisfiable if and only if there is a zero-error triangulation of the corresponding triangulation instance.

References

1 Pankaj K. Agarwal and Subhash Suri. Surface approximation and geometric partitions. In *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '94, page 24–33, USA, 1994. Society for Industrial and Applied Mathematics.

2 Lyuba Alboul, Gertjan Kloosterman, Cornelis Traas, and Ruud van Damme. Best data-dependent triangulations. *Journal of Computational and Applied Mathematics*, 119(1):1–12, 2000.

3 Jeffrey L. Brown. Vertex based data dependent triangulations. *Computer Aided Geometric Design*, 8(3):239–251, 1991.

4 Nira Dyn, David Levin, and Samuel Rippa. Data dependent triangulations for piecewise linear interpolation. *IMA Journal of Numerical Analysis*, 10(1):137–154, 1990.

5 Joachim Gudmundsson, Mikael Hammar, and Marc van Kreveld. Higher order Delaunay triangulations. *Computational Geometry*, 23(1):85–98, 2002.

6 Donald E. Knuth and Arvind Raghunathan. The problem of compatible representatives. *SIAM Journal on Discrete Mathematics*, 5(3):422–427, 1992.

7 Charles L. Lawson. Software for C^1 surface interpolation. In John R. Rice, editor, *Mathematical Software*, pages 161–194. Academic Press, 1977.

8 David Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11:329–343, 1982.

9 Wolfgang Mulzer and Günter Rote. Minimum-weight triangulation is NP-hard. *Journal of the ACM*, 55(2):11:1–11:29, 2008.

10 Alina Nitzke, Benjamin Niedermann, Luciana Fenoglio-Marc, Jürgen Kusche, and Jan-Henrik Haunert. Reconstructing the dynamic sea surface from tide gauge records using optimal data-dependent triangulations. *ARXIVCS*, abs/2009.01012v2, 2020.

11 Samuel Rippa. Minimal roughness property of the delaunay triangulation. *Computer Aided Geometric Design*, 7(6):489–497, 1990.

12 Kai Wang, Chor-Pang Lo, George A. Brook, and Hamid R. Arabnia. Comparison of existing triangulation methods for regularly and irregularly spaced height fields. *International Journal of Geographical Information Science*, 15(8):743–762, 2001.

On the Queue-Number of Partial Orders*

Stefan Felsner[1], Torsten Ueckerdt[2], and Kaja Wille[1]

1 **Institut für Mathematik,**
Technische Universität Berlin, Germany
felsner@math.tu-berlin.de, wille@campus.tu-berlin.de

2 **Institute of Theoretical Informatics**
Karlsruhe Institute of Technology
torsten.ueckerdt@kit.edu

Abstract

The queue-number of a poset is the queue-number of its cover graph viewed as a directed acyclic graph, i.e., the vertex order must be a linear extension of the poset. Heath and Pemmaraju conjectured that every poset of width w has queue-number at most w. Recently, Alam et al. constructed posets of width w with queue-number $w + 1$. Our contribution is a construction of posets with width w with queue-number $\Omega(w^2)$. This (asymptotically) matches the known upper bound.

1 Introduction

A *queue layout* of a graph consists of a total ordering on its vertices and a partition of its edge set into queues, i.e., no two edges in a single block of the partition are nested. The minimum number of queues needed in a queue layout of a graph G is its queue-number and denoted by $\mathsf{qn}(G)$.

To be more precise, let G be a graph and let L be a linear order of the vertices. A *k-rainbow* is a set of k edges $\{a_ib_i : 1 \leq i \leq k\}$ such that $a_1 < a_2 < \cdots < a_k < b_k < \cdots < b_2 < b_1$ in L. A pair of edges forming a 2-rainbow is said to be *nested*. A *queue* is a set of edges without nesting. Given G and L, the edges of G can be partitioned into k queues if and only if there is no rainbow of size $k + 1$ in L. The *queue-number* of G is the minimum number of queues needed to partition the edges of G over all linear orders L.

The queue-number was introduced by Heath and Rosenberg in 1992 [5] as a counterpart of book embeddings. Queue layouts were implicitly used before and have applications in fault-tolerant processing, sorting with parallel queues, matrix computations, scheduling parallel processes, and in communication management in distributed algorithm (see [3,5,7]). There is a rich literature exploring bounds on the queue-number of different classes of graphs [2,3,5,8].

In this note we study the queue-number of posets. This parameter was introduced in 1997 by Heath and Pemmaraju [4]. In the spirit of the older concept of the queue-number of directed acyclic graphs, it is required that a precedes b in a queue layout whenever there is a directed edge $a \to b$, i.e., the queue layout of a directed acyclic graph is a topological ordering and in the case of a poset, it is a linear extension.

A poset P is uniquely characterized by any digraph whose edge set is between the directed cover graph and the directed comparability graph of P. These two digraphs are respectively the transitive reduction and the transitive closure of any digraph representing P. In the context of drawings, embeddings and layouts, it is natural to work with the sparse cover graphs. For example a *diagram* of P is an upward drawing of the directed cover graph.

* S. Felsner was supported by DFG Grant FE 340/13-1.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

The *queue-number* of P, denoted by $\mathsf{qn}(P)$, is the smallest k such that there is a linear extension L of P for which the resulting linear layout of the cover graph G_P contains no $(k+1)$-rainbow. Figure 1 shows an example.

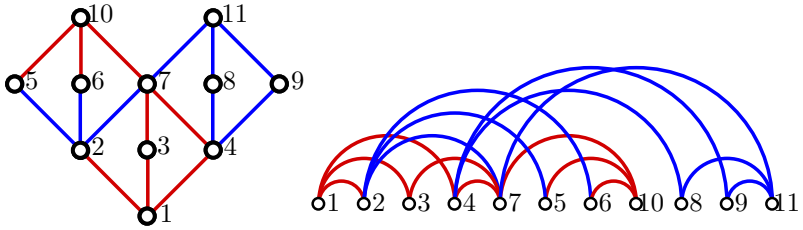


Figure 1 A poset of width 5 and a queue layout with 2 queues indicated by colors.

Clearly $\mathsf{qn}(G_P) \leq \mathsf{qn}(P)$, i.e., the queue-number of a poset is at least as large as the queue-number of its (undirected) cover graph. It was shown by Heath and Pemmaraju [4] that even for planar posets P there is no function f such that $\mathsf{qn}(P) \leq f(\mathsf{qn}(G_P))$. They also investigated the maximum queue-number of several classes of posets, in particular with respect to bounded width (the maximum number of pairwise incomparable elements) and height (the maximum number of pairwise comparable elements). In particular they gave a nice argument showing that $\mathsf{qn}(P) \leq \mathsf{width}(P)^2$ (see Proposition 1 below). The poset P of height 2 and width w whose cover graph is the complete bipartite graph $K_{w,w}$ attains $\mathsf{qn}(P) = \mathsf{width}(P)$. Actually, Heath and Pemmaraju conjectured that $\mathsf{qn}(P) \leq \mathsf{width}(P)$ for every poset P.

Knauer, Micek, and Ueckerdt [6] showed that the inequality $\mathsf{qn}(P) \leq \mathsf{width}(P)$ holds for all posets of width 2. Last year Alam et al. [1] constructed a non-planar poset of width 3 whose queue number is 4. They generalized the example and constructed for every $w > 3$ a poset P with $\mathsf{width}(P) = w$ and $\mathsf{qn}(P) = w+1$. A second contribution of Alam et al. consists in a slight improvement of the upper bound: They show $\mathsf{qn}(P) \leq (w-1)^2+1$ for all posets P of width at most w.

Our contribution is the following theorem.

▶ **Theorem 1.1.** *For every $w > 3$ there is a poset P_w of width w with*

$$\mathsf{qn}(P_w) \geq w^2/8.$$

These examples (asymptotically) match the upper bound. Besides yielding a strong improvement of the lower bound, we also believe that our construction is conceptually simpler than the examples provided by Alam et al. to disprove the conjecture of Heath and Pemmaraju.

As an open problem we promote the question whether the original conjecture holds for planar posets. In [6] it was shown that the queue-number of planar posets of width w is upper bounded by $3w - 2$ and that there are such planar posets P with $\mathsf{qn}(P) = \mathsf{width}(P) = w$.

2 Preliminaries

Before getting serious with our construction, we revisit the nice upper bound argument of Heath and Pemmaraju. Let $P = (X, <)$ be a poset of width w. Dilworth's Theorem asserts that X can be decomposed into w chains of P.

▶ **Proposition 1** (Heath and Pemmaraju). *For every poset P we have $\mathsf{qn}(P) \leq \mathsf{width}(P)^2$.*

Proof. Let $w = \mathsf{width}(P)$, let $C_1, \ldots, C_w$ be a chain partition, and let L be any linear extension of P. Partition the edges of the cover graph into w^2 sets $Q_{i,j}$ with $i, j \in [w]$ such that $(u, v) \in Q_{i,j}$ if $u \in C_i$ and $v \in C_j$. We claim that each $Q_{i,j}$ is a queue.

Let $a < b < c < d$ in L support a pair of nesting cover edges and suppose that both edges (a, d) and (b, c) belong to $Q_{i,j}$. By definition $a, b \in C_i$ and $c, d \in C_j$ and from the ordering in L we get $a < b$ and $c < d$ in P. Now we have $a < b$ and $b < c$ and $c < d$ in P whence the relation $a < d$ is implied by transitivity. This contradicts that (a, d) is a cover edge. ◀

In fact we have shown a much stronger statement: If P and a chain partition $C_1, \ldots, C_w$ are given, then there is a partition of the edges of the cover graph of P into parts $Q_{i,j}$ with $i, j \in [w]$ such that each $Q_{i,j}$ is a queue for every linear extension L of P.

2.1 Concepts needed for the construction

Let P be a poset. The *dual* of P, denoted $\bar{P}$, is the poset on the same ground set such that: $x < y$ in $P \iff y < x$ in $\bar{P}$.

A poset P is *2-dimensional* if and only if there are two linear extensions L_1 and L_2 such that: $x < y$ in $P \iff x < y$ in L_1 and L_2. Such a pair L_1, L_2 is called a *realizer* of P.

When drawing 2-dimensional posets, it is common to represent each element x by a point with coordinates (x_1, x_2) where x_1 is the position of x in L_1 and x_2 is the position of x in L_2.

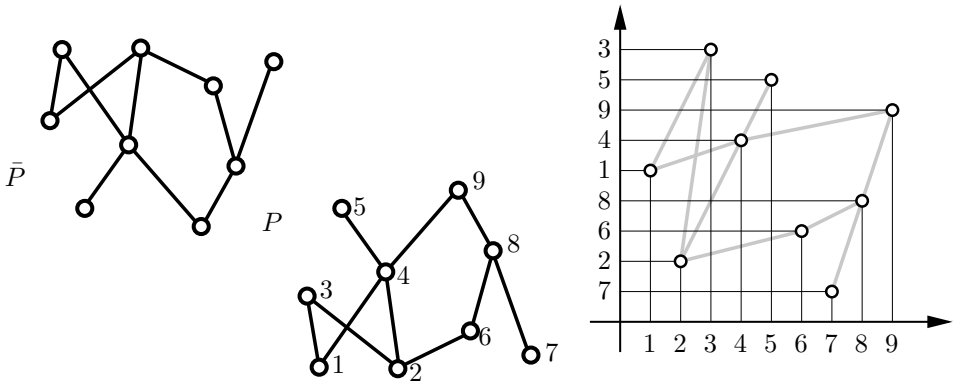


Figure 2 A poset P, its dual $\bar{P}$, and a 2-dimensional drawing of P.

3 Proof of Theorem 1.1

We define P_w recursively. For $w = 1, 2$ we let P_w be a w-chain, for $w \geq 3$, the construction of P_w is based on a copy of P_{w-2}, a reinforcement poset R_{w-2} of width $w-2$ with two linear extensions L_x and L_y, the duals $\bar{P}_{w-2}$ and $\bar{R}_{w-2}$ of P_{w-2} and R_{w-2}, and two additional points a and b. Due to space limitations, we refrain from giving a full formal description of the construction. Instead, we invite the reader to take a look at Figure 3, which shows a sketch of P_w. Note that the number $p(w)$ of vertices of P_w is given by the recursion $p(w) = 2p(w-2) + 6r(w-2) + 2$, where $r(w)$ is the number of vertices of R_w. The edges between X and R_{w-2} are given by L_x such that for each i the i-th element of the chain X is connected to the element of R_{w-2} which is at position i in L_x. The edges between Y and R_{w-2} are given by L_y in the same way.

It can be seen from the sketch that P_w is self-dual, the reflection $P_w \leftrightarrow \bar{P}_w$ has two fixed points a and b. This shows that when analyzing $\mathsf{qn}(P_w)$, we can restrict the attention to

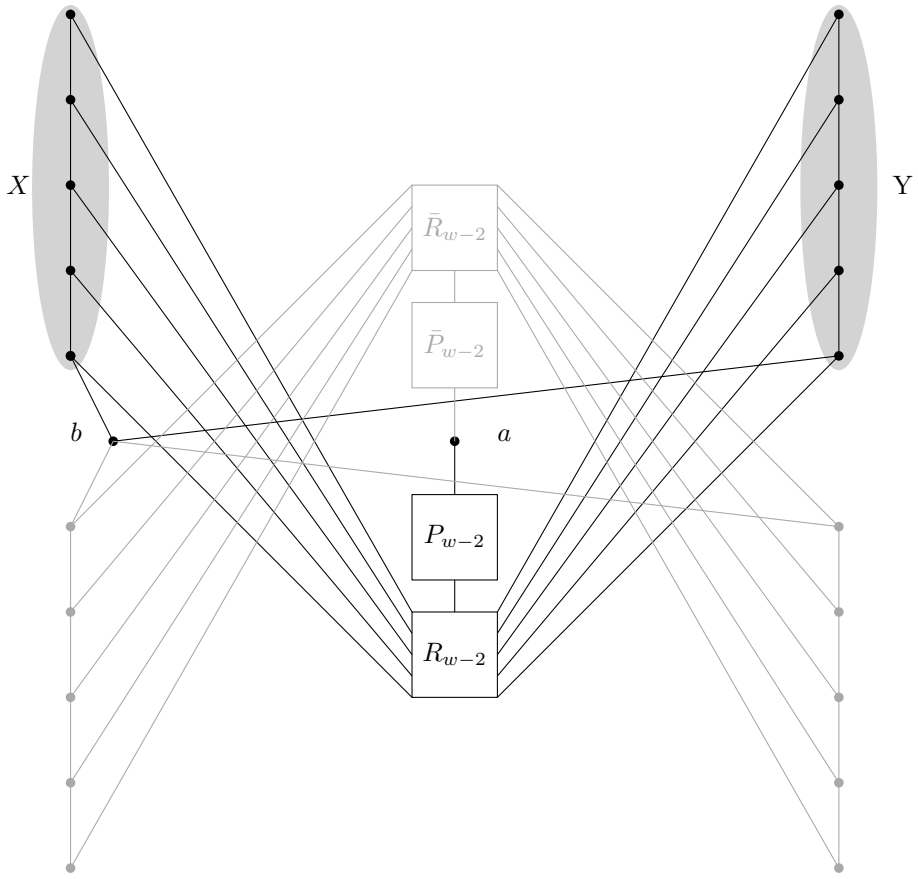


Figure 3 Recursive construction of P_w.

linear extensions of P_w which have a before b. With this assumption, a rainbow between R_{w-2} and either X or Y nests above each rainbow of P_{w-2}. If we let q_{w-2} be the size of a rainbow between R_{w-2} and either X or Y, then we have the recursion:

$$\mathsf{qn}(P_w) \geq \mathsf{qn}(P_{w-2}) + q_{w-2} \tag{1}$$

We think of this use of a self-dual construction as the *symmetry trick*. Constructions given in [6] (Proof of Prop. 2) and [1] (Proof of Thm. 4) also use a recursion based on two copies of the poset from the previous level of the recursion. This allows them to add an edge nesting over the rainbow from the previous level of the recursion.

Now suppose that for each width $u < w$, we choose the poset R_u to be an antichain of size u and the linear extensions L_x and L_y to be a realizer (think of L_x as the identity permutation and of L_y as its reverse). The Lemma of Erdős-Szekeres asserts that in every linear extension of R_u there is an increasing or a decreasing sequence of size at least $\lceil\sqrt{u}\rceil$, i.e., $q_u = \lceil\sqrt{u}\rceil$.

This value of q_u together with inequality (1) yields

$$\mathsf{qn}(P_w) \geq \sum_{u<w;\ u\equiv w(2)} \left\lceil\sqrt{u}\right\rceil \quad \in \Theta(w^{3/2}).$$

For the proof of the theorem we need a better construction for the reinforcement posets R_u. Such a construction will be provided in the proof of the following lemma[1].

▶ **Lemma 3.1.** *For each $u \geq 1$, there is a 2-dimensional poset R_u of width u with a realizer L_x, L_y, such that if L is a linear extension of R_u and d_x and d_y denote the maximum lengths of an increasing sequence of L which is decreasing in L_x and L_y respectively, then $d_x + d_y \geq u + 1$.*

The lemma says that we can assume the value $q_u = \lceil \frac{u+1}{2} \rceil$. With inequality (1) we get:

$$\mathsf{qn}(P_w) \geq \sum_{u<w;\ u\equiv w(2)} \left\lceil \frac{u+1}{2} \right\rceil$$

In the case w odd, $w = 2s+1$, we get $\mathsf{qn}(P_w) \geq \sum_{k=1}^{s} k = \binom{s+1}{2}$. In the case w even, $w = 2s$, we get $\mathsf{qn}(P_w) \geq \sum_{k=2}^{s} k = \binom{s+1}{2} - 1$. A simple computation shows that for $w \geq 4$ we get $\mathsf{qn}(P_w) \geq w^2/8$, independent of the parity of w. This completes the proof.

3.1 The construction of R_u for Lemma 3.1

The construction of R_u is again recursive. Let R_1 be a single point. Then clearly $d_x + d_y = 2$. For the construction of R_u for $u \geq 2$ we again use the symmetry trick. We take a series composition of $Q_1 + a + Q_2$ where Q_1 and Q_2 are two copies of R_{u-1} and compose it in parallel with a single element b. We remark that element a is here only for the sake of the exposition. The choice of the realizer L_x, L_y is depicted in Figure 4. Clearly $\mathsf{width}(R_u) = \mathsf{width}(R_{u-1}) + 1 = u$.

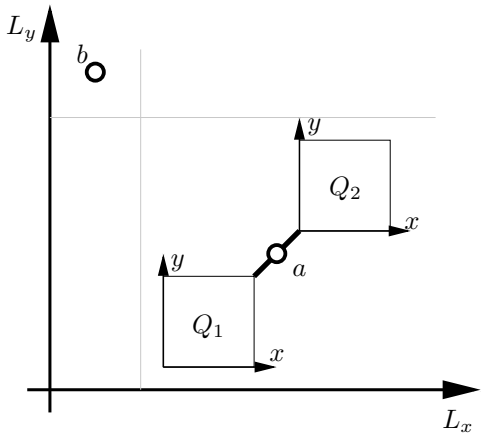


Figure 4 The recursive construction of R_u with its realizer L_x, L_y.

Let L be any linear extension of R_u. First suppose that $a < b$ in L. Let L' be the restriction of L to Q_1. By induction the lengths d'_x and d'_y of increasing sequences of L' which are decreasing in the two linear extensions of the realizer of Q_1 satisfy $d'_x + d'_y \geq u$. Since b precedes Q_1 in L_x and comes after Q_1 in L, we have $d_x \geq d'_x + 1$. Together with the trivial $d_y \geq d'_y$, we get $d_x + d_y \geq u + 1$.

[1] The lemma with a different proof was originally discovered by the first and the second author in joint research with Francois Dross, Piotr Micek, and Michał Pilipczuk.

If $b < a$ we consider Q_2. As before we get the two values d'_x and d'_y for the restriction L' of L to Q_2 and know by induction that $d'_x + d'_y \geq u$. The position of b relative to Q_2 in L and L_y shows that $d_y \geq d'_y + 1$, whence again $d_x + d_y \geq u + 1$. This completes the proof of the lemma.

4 Conclusion

We have made substantial progress in the understanding of queue-numbers of partial ordered sets. We take the opportunity to list and comment on open questions in the field.

- An obvious question is to ask for improved upper and lower bounds. More precisely, we now know that the growth rate of the queue-number of posets of width w is $(C+o(1))w^2$ for some constant C between $1/8$ and 1. What is the precise value of constant C?
- What is the maximum queue-number of planer posets of width w? Knauer, Micek, and Ueckerdt [6] proved the lower bound w and the upper bound $3w - 2$.
- Heath and Pemmaraju [4] conjectured that planar posets on n elements have queue-number at most $\sqrt{n}$. The k antichain together with a matching up to a chain X and a matching down to a chain Y such that the chains represent a dual pair of linear extensions is a planar poset P with width $n = 3k$ elements and $\mathsf{qn}(P) = \sqrt{\lceil n/3 \rceil}$.
- In [6] is was shown that posets P of width 2 have $\mathsf{qn}(P) \leq 2$. In [1] it was shown that posets P of width 3 may have $\mathsf{qn}(P) \geq 4$ and satisfy $\mathsf{qn}(P) \leq 5$. Is 4 or 5 the best upper bound in this case?

References

1 J. M. Alam, M. A. Bekos, M. Gronemann, M. Kaufmann, and S. Pupyrev, *Lazy queue layouts of posets*, arXiv, 2008.10336 (2020). To appear in Proc. GD 2020.

2 V. Dujmović, G. Joret, P. Micek, P. Morin, T. Ueckerdt, and D. R. Wood, *Planar graphs have bounded queue-number*, Journal of the ACM, 67 (2020), 22:1–22:38.

3 L. S. Heath, F. T. Leighton, and A. L. Rosenberg, *Comparing queues and stacks as machines for laying out graphs*, SIAM Journal on Discrete Mathematics, 5 (1992), 398–412.

4 L. S. Heath and S. V. Pemmaraju, *Stack and queue layouts of posets*, SIAM Journal on Discrete Mathematics, 10 (1997), 599–625.

5 L. S. Heath and A. L. Rosenberg, *Laying out graphs using queues*, SIAM Journal on Computing, 21 (1992), 927–958.

6 K. Knauer, P. Micek, and T. Ueckerdt, *The queue-number of posets of bounded width or height*, in Proc. GD 2018, vol. 11282 of LNCS, Springer, 2018, pp. 200–212.

7 J. Nešetřil, P. Ossona de Mendez, and D. R. Wood, *Characterisations and examples of graph classes with bounded expansion*, European J. Combin., 33 (2012), 350–373.

8 V. Wiechert, *On the queue-number of graphs with bounded tree-width*, The Electronic Journal of Combinatorics, 24 (2017), 1–65.

Uncertainty-Region Lower Bounds for the Preprocessing Model of Uncertainty

Ivor van der Hoog[1], Maarten Löffler[1], Irina Kostitsyna[2], and Bettina Speckmann[2]

1 **Department of Information and Computing Sciences, Utrecht University, the Netherlands**
2 **Department of Mathematics and Computer Science, TU Eindhoven, the Netherlands**

Abstract

In the preprocessing model for uncertain data we are given a set of regions $\mathcal{R}$ which model the uncertainty associated with an unknown set of points P. In this model there are two phases: a preprocessing phase, in which we have access only to $\mathcal{R}$, followed by a reconstruction phase, in which we have access to points in P at a certain retrieval cost per point. We study the following algorithmic question: how fast can we construct the Pareto front of P in the preprocessing model?

We introduce a new concept for algorithmic efficiency in the preprocessing model by introducing the *uncertainty region lower bound*. We then apply this lower bound to the Pareto front.

Related Version https://arxiv.org/abs/2101.06079

1 Introduction

In many applications of geometric algorithms the input is inherently imprecise. A classic example are GPS samples, which have a significant error. Geometric imprecision can be caused by other factors as well. For example, when measuring a moving object, its location may have an error dependent on its speed. Another example comes from I/O-sensitive computations: exact locations may be too costly to store in local memory [3]. Algorithms that can handle imprecise input have received considerable attention in computational geometry.

Preprocessing model. Held and Mitchell [14] introduced the *preprocessing model of uncertainty* as a model to study the amount of geometric information contained in uncertain points. In this model, the input is a set of geometric (uncertainty) regions $\mathcal{R} = (R_1, R_2, \ldots, R_n)$ with an associated "true" planar point set $P = (p_1, p_2, \ldots, p_n)$. For any pair $(\mathcal{R}, P)$, we say that P *respects* $\mathcal{R}$ if each p_i lies inside its associated region R_i; we assume throughout the paper that P respects $\mathcal{R}$. The preprocessing model has two consecutive phases: a preprocessing phase where we have access only to the set of uncertainty regions $\mathcal{R}$ and a reconstruction phase where we can for each $R_i \in \mathcal{R}$, retrieve the true location p_i. We want to preprocess $\mathcal{R}$ to create some linear-size auxiliary datastructure Ξ. Afterwards, we want to reconstruct the desired output on P using Ξ faster than without preprocessing.

Löffler and Snoeyink [16] were the first to interpret $\mathcal{R}$ as a collection of imprecise measurements of a true point set P. The size of Ξ and the running time of the reconstruction phase, together quantify the information about (the Delaunay triangulation of) P contained in $\mathcal{R}$. This interpretation was widely adopted within computational geometry and motivated recent results for constructing Delaunay triangulations [4, 5, 9, 20], convex hulls [12, 13, 17, 18] and other planar decompositions [15, 19] for imprecise points. Bruce *et al.* [3] study a problem related to the prepreocessing model where they attach a fixed cost C to each retrieval. We adopt their approach, but remove their assumption that C is a constant.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

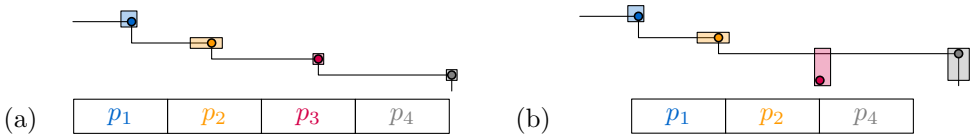


Figure 1 The Pareto front of P can be implied by the geometry of $\mathcal{R}$ (left) or not (right).

Output format. Classical work in the preprocessing model ultimately aims to preprocess the data in such a way that one can achieve a (near-)*linear*-time reconstruction phase. Indeed, if the final output structure has linear complexity and must explicitly contain the coordinates of each value in P, then returning the result takes trivially at least $\Omega(n)$ time. However, this point of view is limiting in two ways. First, certain geometric problems, such as the convex hull or the Pareto front, may have sub-linear output complexity. Second, even if the output has linear complexity, it may be possible to find its combinatorial structure without inspecting the true locations of all points. Consider the example in Figure 1: on the left, we do not need to retrieve any point; on the right, we do not need to retrieve p_3 after we retrieve p_4. Van der Hoog *et al.* [19] propose an addition to the preprocessing model to enable a more fine-grained analysis in these situations: instead of returning the desired structure on P explicitly, they instead return an *indirect representation* of the output. This indirect representation can take the form of a pointer structure which is isomorphic to the desired output on P, but where each value is a pointer to either a certain (retrieved) point, or to an uncertain (unretrieved) point.

Contribution. We define a new concept for algorithmic efficiency for constructing an indirect representation in the preprocessing model. Specifically, in Section 2 we introduce the idea of an *uncertainty-region lower bound*. We show in Section 3 how to use this idea to obtain a non-trivial lower bound for the construction of the Pareto front of imprecise points. In the full version, we also present an algorithm which matches the lower bound.

2 Algorithmic lower bounds

To assess the efficiency of any algorithm we generally want to compare its performance to a suitable lower bound. To lower bound the running time of an algorithmic problem, one must first reason about the computational model in which the problem is studied. In the full version, we elaborately discuss each well-studied computational model and reason about how each of these apply. Henceforth, we make the most general assumption: which is that we run our algorithm on a real-valued, comparison-based pointer machine. Two common types of lower bounds are *worst-case* and *instance* lower bounds. The classical worst-case lower bound takes the minimum over all algorithms A, of the maximal running time of A for any pair $(\mathcal{R}, P)$. The instance lower bound [1, 11] is the minimum over all A, for a fixed instance $(\mathcal{R}, P)$, of the running time of A on $(\mathcal{R}, P)$. For the Pareto front the worst-case lower bound is trivially $\Omega(n)$; worst-case optimal performance (for us, in the reconstruction phase) is hence easily obtainable. Instance-optimality, on the other hand, is unobtainable in classical computational geometry [1]. Consider, for example, the searching problem for a value q amongst a set X of sorted numbers. For each instance (X, q), there exists a naive algorithm that guesses the correct answer in constant time. Thus the instance lower bound for binary search is constant, even though there is no algorithm that can perform binary search in constant time in a comparison-based RAM model [10]. Hence we introduce a new

lower bound for the preprocessing model, whose granularity falls in between the instance and worst-case lower bound. Our *uncertainty-region* lower bound is the minimum over all A, for a fixed input $\mathcal{R}$, of the maximal running time of A on $(\mathcal{R}, P)$ for any P that respects $\mathcal{R}$. The full version extensively reviews lower bound definitions.

We briefly revisit the definitions of worst-case and instance lower bounds in the preprocessing model and then formally introduce our new uncertainty-region lower bound.

Worst-case lower bounds. The worst-case comparison-based lower bound of an algorithmic problem $\mathcal{P}$ considers each algorithm[1] plus datastructure pair (A, Ξ) which solves $\mathcal{P}$ in a *competitive setting* with respect to their maximal running time:

$$\text{Worst-case lower bound}(\mathcal{P}) := \min_{(A,\Xi)} \max_{(\mathcal{R},P)} \text{Runtime}(A, \Xi, \mathcal{R}, P)\,.$$

At this point, we wish to note that in classical algorithms, the worst-case lower bound is the minimum runtime over all algorithms. In the reconstruction phase of the preprocessing model, we take the minimum over all pairs of algorithms A and auxiliary datastructures Ξ; since we want to include the preprocessing done in the preprocessing phase. Let L be the number of distinct outcomes for all instances $(\mathcal{R}, P)$ implies a lower bound on the maximal running time for any algorithm A: regardless of preprocessing, auxiliary datastructures and memory used, any comparison-based pointer machine algorithm A can be represented as a decision tree where at each algorithmic step, a binary decision is taken [2, 7, 10]. Since there are at least L different outcomes, there must exists a pair $(\mathcal{R}, P)$ for which A takes $\log L$ steps before A terminates (this lower bound is often referred to as the *information theoretic lower bound* or sometimes the *entropy* of the problem [1, 6, 7], full version).

Instance lower bounds. A stronger lower bound, is an instance lower bound [1]. For a given instance $(\mathcal{R}, P)$, the instance lower bound of the reconstruction phase is:

$$\text{Instance lower bound}(\mathcal{P}, \mathcal{R}, P) = \min_{(A,\Xi)} \text{Runtime}(A, \Xi, \mathcal{R}, P)\,.$$

Note that, just as with the worst-case lower bound, classical instance lower bounds ([1]) minimize over all algorithms A. We minimize over all pairs $(A\Xi)$ to translate their definition to the preprocessing model. An algorithm A is instance optimal, if for every instance $(\mathcal{R}, P)$ the runtime of A matches the instance lower bound. Löffler *et al.* [15] define *proximity structures* that include quadtrees, Delaunay triangulations, convex hulls, Pareto fronts and Euclidean minimum spanning trees.

[1] We refer to comparison-based algorithms algorithms on an intuitive level: as RAM computations that do not make use of flooring. For a more formal definition we refer to any of [1, 2, 8, 10].

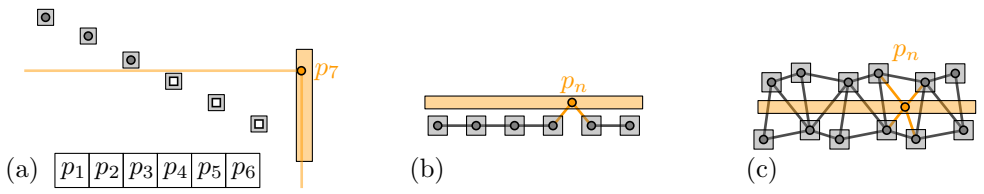


Figure 2 Thrice a collection of grey uncertainty regions where the Pareto front, EMST or Delaunay triangulation of the grey points is implied by the regions; plus an orange region R_n. Depending on the placement of p_n, it can neighbor any grey point in the final structure.

▶ **Theorem 1.** *[Illustrated by Figure 2.] Let the unspecified retrieval cost C not dominate $O(\log n)$ RAM instructions and $\mathcal{R}$ be any set of pairwise disjoint uncertainty rectangles. Then there exists no algorithm A in the preprocessing model with indirect representation that can construct a proximity data structure on the true points which is instance optimal.*

Uncertainty-region lower bounds. Worst-case optimality is easily attainable and we proved that instance optimality is not attainable in the preprocessing model. Yet the examples in Figure 1 and 2 intuitively have a lower bound of $\Theta(1)$ and $\Theta(\log n + C)$, which is trivial to match. We capture this intuition for an algorithmic problem $\mathcal{P}$ with input $\mathcal{R}$:

$$\text{Uncertainty-region lower bound}(\mathcal{P}, \mathcal{R}) := \min_{(A,\Xi)} \max_{(P \text{ respects } \mathcal{R})} \text{Runtime}(A, \Xi, \mathcal{R}, P)\,,$$

and say an algorithm A is uncertainty-region optimal if for every $\mathcal{R}$, A has a running time that matches the uncertainty-region lower bound. Denote by $L(\mathcal{R})$ the number of distinct outcomes for all P that respect $\mathcal{R}$. Via the information theoretic lower bound we know:

$$\forall \mathcal{R}, \quad \log |L(\mathcal{R})| \leq \text{Uncertainty-region lower bound}(\mathcal{P}, \mathcal{R})\,.$$

For constructing proximity structures in the preprocessing model with indirect representations, the value of $\log L(\mathcal{R})$ can range from anywhere between 0 and $n \log n$. Consequently, an optimal algorithm cannot necessarily afford to explicitly retrieve the entire point set P.

3 The Pareto front

Let $\mathcal{R} = (R_1, R_2, \ldots, R_n)$ be a sequence of n pairwise disjoint closed axis-aligned uncertainty rectangles, with underlying point set P. For ease of exposition, we assume $\mathcal{R}$ and P lie in general position (no points or region vertices share a coordinate). We denote by $[R_i, R_j] := (R_i, R_{i+1}, \ldots, R_j)$ a subsequence of $j - i + 1$ regions and similarly by $[p_i, p_j]$ a subsequence of points. For two points p and q, we say that p *(Pareto) dominates* q if both its x- and y-coordinates are greater than or equal to the respective coordinates of q. We say a point p dominates a region R if it dominates the top right vertex. We define the *Pareto front* of P as the boundary of the set of points that are dominated by a point in P. That is, the Pareto front is the set of points in P that are not dominated by any other point in P, connected by a rectilinear staircase. For any region or point R, we define its *horizontal halfslab* as the union of all horizontal halflines that are directed leftward, whose apex lies in or on R. We define the *vertical halfslab* symmetrically using downward vertical halflines.

In this section, we construct a function $CP(\mathcal{R}, P)$ that for any pair $(\mathcal{R}, P)$ outputs a value. Given $\mathcal{R}$, for each P we might obtain a different value $CP(\mathcal{R}, P)$. However we show that all of these values lower bound $\log |L(\mathcal{R})|$ and thus Uncertainty-region lower bound$(\mathcal{P}, \mathcal{R})$.

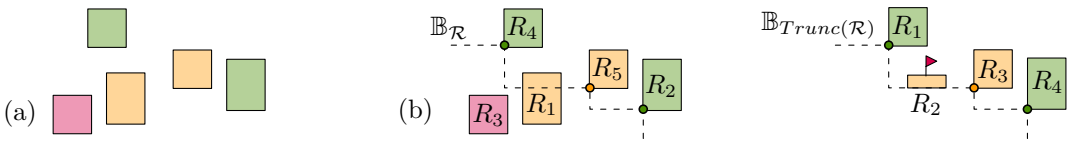


Figure 3 Left: a collection of uncertainty regions. Green is positive, red is negative and yellow is potential. The horizontal halfslab of a green region is shown. Right: A collection of uncertainty regions before and after truncation, note that we re-indexed the regions and flagged one.

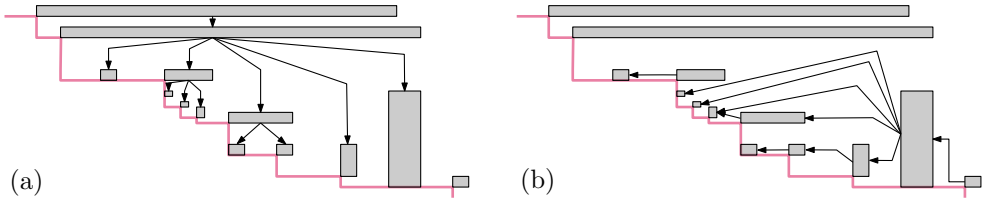


Figure 4 A truncated set and its horizontal and vertical arrows.

We define a procedure *Trunc*. Given an *original* set $\mathcal{R}$ of n pairwise disjoint axis-aligned rectangles, $Trunc(\mathcal{R})$ returns a *truncated set* $\mathcal{R}'$ where some regions may be *flagged* (marked with a boolean). Refer to Figure 3. Specifically, all area from $\mathcal{R}$, that is dominated by *any* bottom left vertex in $\mathcal{R}$ gets removed. The remaining set of rectangular regions is the set $\mathcal{R}' = Trunc(\mathcal{R})$. Each region in $\mathcal{R}'$ that is not empty, but different from its corresponding region in $\mathcal{R}$ gets *flagged*. Observe that the Pareto front of the bottom left vertices of $\mathcal{R}$ induce a well-defined order *Trunc* re-indexes the remaining regions according to top left to bottom right ordering of their bottom left vertices.

Dependency graphs. Given a truncated set $\mathcal{R}$, we define a *(directed) dependency graph* denoted by $G(\mathcal{R})$ as follows. The nodes of the graph correspond to regions in $\mathcal{R}$. We have two types of directed edges (horizontal and vertical arrows). A region R_i has a *vertical arrow* to R_j if R_j *succeeds* R_i and is vertically visible from R_i (there exists a vertical segment connecting R_i and R_j that does not intersect any other region in $\mathcal{R}$). A region R_i has a *horizontal arrow* to R_j if R_j *precedes* R_i and is horizontally visible from R_i. Refer to Figure 4.

The Pareto cost function. Now we are almost ready to define the function $\mathrm{CP}(\mathcal{R}, P)$ for truncated sets $\mathcal{R}$. Before we can define the Pareto cost function, we define additional concepts (Figure 5). By C we denote the unspecified cost for a retrieval. For all regions $R_i \in \mathcal{R}$, we denote by V_i the subset of $[R_i, R_n]$ that is vertically visible from R_i (including R_i itself) and by H_i the subset of $[R_1, R_i]$ that is horizontally visible from R_i (including R_i itself). Given P, we denote by $V_i(P) \subseteq V_i$: the union of $\{R_i\}$ with the subset of V_i of regions that are dominated by a point p_j with $j \leq i$. The set $H_i(P)$ is defined symmetrically.

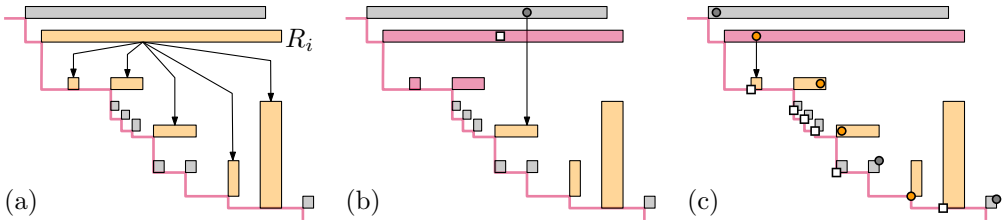


Figure 5 A region R_i and the set V_i in orange. Middle: for a given set of points, the set $V_i(P)$ is shown in red. Right: the set $V_i(P)$ changes for different P, but always includes R_i.

Intuitively, the truncation operator represents the foresight about the Pareto front of P. Now, given a truncated set $\mathcal{R}$ and P we construct a set $\tilde{\mathcal{R}}(P) \subset \mathcal{R}$ that intuitively represents which regions of $\mathcal{R}$ were geometrically interesting in hindsight. $\tilde{\mathcal{R}}(P)$ is the subset of $\mathcal{R}$ where each $R_i \in \tilde{\mathcal{R}}(P)$ is intersected by the Pareto front of P *and* one of three conditions holds:

1. R_i is flagged;
2. R_i intersects and edge e with endpoint $p_j \in P$ and $i \neq j$; and/or

3. R_i is not a sink in $G(\mathcal{R})$.

We define the *Pareto cost function* as: $\mathrm{CP}(\mathcal{R}, P) = \sum_{R_i \in \tilde{\mathcal{R}}(P)} C + \log|V_i(P)| + \log|H_i(P)|$.

Reconstruction. In the reconstruction phase an algorithm can use *any* auxiliary structure Ξ to aid its computation. In the remainder of this section we consider *any* truncated set $\mathcal{R}$ of n elements, together with *any* auxiliary datastructure. We provide an information-theoretical lower bound, which depends on $\mathcal{R}$ and P, for both the number of RAM instructions and disk retrievals required to construct an indirect output representation Ξ^* regardless of Ξ.

3.1 A lower bound for disk retrievals

In the full version we revisit the instance-based lower bound proof by Bruce *et al.* [3] to obtain the lemma below. The prior result by Bruce *et al.* enables one to identify for each set of regions, a triple of uncertainty regions for which at least one point needs to be retrieved (at cost unknown C). The authors then retrieve all three points, remove all areas dominated by a point, and recurse. The novelty in our argument lies in the fact that for each pair $(\mathcal{R}, P)$ we can immediately, non-recursively characterize the regions which require a disk retrieval using $\tilde{\mathcal{R}}(P)$ which is a prerequisite for creating a data structure that will return these regions.

▶ **Lemma 2.** *Let $\mathcal{R}$ be a truncated set and let P be any point set that respects $\mathcal{R}$. Any algorithm that constructs Ξ^* of P must perform at least $\frac{1}{3}|\tilde{\mathcal{R}}(P)|$ retrievals at cost C each.*

3.2 A lower bound on RAM instructions

In Section 2 we defined the uncertainty-region lower bound. By an information-theoretical lower bound (algebraic decision tree or entropy [1, 7]), we have, for any $\mathcal{R}$, that the Uncertainty-region lower bound is at least $\log L(\mathcal{R})$, where $L(\mathcal{R})$ is the number of combinatorially different Pareto fronts of point sets that respect $\mathcal{R}$. We prove the following:

▶ **Lemma 3.** *Let $\mathcal{R}$ be a truncated set and P be any point set that respects $\mathcal{R}$. Then*

$$\sum_{R_i \in \tilde{\mathcal{R}}(P)} \log|V_i(P)| + \log|H_i(P)| \leq 2 \cdot \log L(\mathcal{R})\,.$$

Given Lemma 2 and Lemma 3 we can immediately conclude:

▶ **Theorem 4.** *Let $\mathcal{R}$ be a truncated set and P be* any *set that respects $\mathcal{R}$. Then* $\mathrm{CP}(\mathcal{R}, P)$ *is fewer than three times the uncertainty-region lower bound of $\mathcal{R}$.*

In the full version, we show that for each pair $(\mathcal{R}, P)$ this lower bound can be matched.

References

1 Peyman Afshani, Jérémy Barbay, and Timothy M. Chan. Instance-optimal Geometric Algorithms. *Journal of the ACM (JACM)*, 64(1):1–38, 2017.

2 Michael Ben-Or. Lower Bounds for Algebraic Computation Trees. In *Proceedings ACM Symposium on Theory of Computing*, pages 80–86, 1983.

3 Richard Bruce, Michael Hoffmann, Danny Krizanc, and Rajeev Raman. Efficient Update Strategies for Geometric Computing with Uncertainty. *Theory of Computing Systems*, 38(4):411–423, 2005.

4 Kevin Buchin, Maarten Löffler, Pat Morin, and Wolfgang Mulzer. Delaunay Triangulation of Imprecise Points Simplified and Extended. *Algorithmica*, 61:674–693, 2011.

5 Kevin Buchin and Wolfgang Mulzer. Delaunay Triangulations in $O(sort(n))$ ime and more. *Journal of the ACM (JACM)*, 58(2):6, 2011.

6 Jean Cardinal, Samuel Fiorini, and Gwenaël Joret. Minimum Entropy Coloring. In *International Symposium on Algorithms and Computation (ISAAC)*, pages 819–828, 2005.

7 Jean Cardinal, Gwenaël Joret, and Jérémie Roland. Information-theoretic Lower Bounds for Quantum Sorting. *arXiv preprint:1902.06473*, 2019.

8 Erik D Demaine, Adam C Hesterberg, and Jason S Ku. Finding Closed Quasigeodesics on Convex Polyhedra. In *Symposium on Computational Geometry (SoCG)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

9 Olivier Devillers. Delaunay Triangulation of Imprecise Points: Preprocess and Actually get a Fast Query Time. *Journal of Computational Geometry (JoCG)*, 2(1):30–45, 2011.

10 Jeff Erickson, Ivor van der Hoog, and Tillmann Miltzow. Smoothing the Gap between NP and ER. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, 2020.

11 William Evans, David Kirkpatrick, Maarten Löffler, and Frank Staals. Competitive Query Strategies for Minimising the Ply of the Potential Locations of Moving Points. In *International Symposium on Computational Geometry (SoCG)*, pages 155–164. ACM, 2013.

12 William Evans and Jeff Sember. The Possible Hull of Imprecise Points. In *Canadian Conference on Computational Geometry (CCCG)*, 2011.

13 Esther Ezra and Wolfgang Mulzer. Convex Hull of Points Lying on Lines in $o(n \log n)$ Time after Preprocessing. *Computational Geometry*, 46(4):417–434, 2013.

14 Martin Held and Joseph SB Mitchell. Triangulating Input-constrained Planar Point Sets. *Information Processing Letters*, 109(1):54–56, 2008.

15 Maarten Löffler and Wolfgang Mulzer. Unions of Onions: Preprocessing Imprecise Points for Fast Onion Decomposition. *Journal of Computational Geometry (JoCG)*, 5:1–13, 2014.

16 Maarten Löffler and Jack Snoeyink. Delaunay Triangulation of Imprecise Points in Linear Time after Preprocessing. *Computational Geometry*, 43(3):234–242, 2010.

17 Maarten Löffler and Marc van Kreveld. Largest and Smallest Convex Hulls for Imprecise Points. *Algorithmica*, 56(2):235, 2010.

18 Takayuki Nagai, Seigo Yasutome, and Nobuki Tokura. Convex Hull Problem with Imprecise Input and its Solution . *Systems and Computers in Japan*, 30(3):31–42, 1999.

19 Ivor van der Hoog, Irina Kostitsyna, Maarten Löffler, and Bettina Speckmann. Preprocessing Ambiguous Imprecise Points. In *International Symposium on Computational Geometry (SoCG)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2019.

20 Marc van Kreveld, Maarten Löffler, and Joseph SB Mitchell. Preprocessing Imprecise Points and Splitting Triangulations. *SIAM Journal on Computing*, 39(7):2990–3000, 2010.

Approximating Independent Set and Dominating Set on VPG graphs

Esther Galby[1] and Andrea Munaro[2]

1 **CISPA Helmholtz Center for Information Security, Saarbrücken, Germany**
galby@cispa.saarland

2 **School of Mathematics and Physics, Queen's University Belfast, UK**
a.munaro@qub.ac.uk

Abstract

We consider Independent Set and Dominating Set restricted to VPG graphs. Combining the well-known Baker's shifting technique with bounded mim-width arguments, we provide simple PTASes for these two problems on VPG graphs admitting a representation such that each grid-edge belongs to at most t paths and the length of the horizontal part of each path is at most c, for some $c \geq 1$.

1 Introduction

Given a family O of geometric objects in the plane, the intersection graph of O has the objects in O as its vertices and two vertices $o_i, o_j \in O$ are adjacent in the graph if and only if $o_i \cap o_j \neq \varnothing$. If O is a set of curves in the plane, where a *curve* is a subset of $\mathbb{R}^2$ homeomorphic to the unit interval $[0, 1]$, then the intersection graph of O is a *string graph*. Many important graph classes like planar graphs and chordal graphs are subclasses of string graphs [16, 20] and so it is natural to study classic optimization problems such as Independent Set and Dominating Set on string graphs.

Asinowski et al. [1] introduced the class of *Vertex intersection graphs of Paths on a Grid* (*VPG graphs* for short). A graph is a *VPG graph* if one can associate a path on a grid with each vertex such that two vertices are adjacent if and only if the corresponding paths intersect on at least one grid-point. It is not difficult to see that the class of VPG graphs coincides with that of string graphs [1]. For a VPG graph G, the pair $\mathcal{R} = (\mathcal{G}, \mathcal{P})$ consisting of a rectangular grid $\mathcal{G}$ and a family of paths $\mathcal{P}$ on $\mathcal{G}$ satisfying the property above is a *VPG representation* of G. If a VPG graph admits a representation $\mathcal{R}$ such that each path in $\mathcal{P}$ has at most k *bends* i.e., 90 degrees turns at a grid-point, the graph is a *B_k-VPG graph*. The *length* of a path $P \in \mathcal{P}$ is the number of grid-edges used and a *segment* of P is either a vertical or horizontal line segment in the polygonal curve constituting P.

Independent Set is known to be NP-complete on B_k-VPG graphs even for $k = 0$ [23]. Therefore, there has been a focus on providing approximation algorithms for restricted subclasses of string graphs. Fox and Pach [14] gave, for every $\varepsilon > 0$, a n^{ε}-approximation algorithm for *k-string graphs* i.e., string graphs in which every two curves intersect each other at most k times. Lahiri et al. [24] provided a $O(\log^2 n)$-approximation algorithm for B_1-VPG graphs and a $O(\log d)$-approximation algorithm for equilateral B_1-VPG graphs (a B_1-VPG graph is *equilateral* if, for each path, its horizontal and vertical segment have the same length), where d denotes the ratio between the maximum and minimum length of segments of paths. Finally, they showed that Independent Set on equilateral B_1-VPG graphs where each horizontal and vertical segment have length 1 is NP-complete. Improving on [24], Biedl and Derka [5] provided a $4 \log n$-approximation algorithm for the weighted version of Independent Set on B_2-VPG graphs. In the case of B_1-VPG, this result

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

was further improved by Bose et al. [6], who provided a $4\max\{1, \log \mathsf{OPT}\}$-approximation algori-thm for the weighted version of Independent Set.

Dominating Set is APX-hard on 1-string B_1-VPG graphs (see, e.g., [26]). Mehrabi [26] considered the subclass of 1-string B_1-VPG graphs in which no endpoint of a path belong to any other path and provided a $O(1)$-approximation algorithm. Bandyapadhyay et al. [3] considered intersection graphs of L-frames, where an *L-frame* is a path on a grid with exactly one bend, and provided a $(2+\varepsilon)$-approximation algorithm in the case the bend of each path belongs to a diagonal line with slope -1. They also showed that the problem is APX-hard if each L-frame intersects a diagonal line and that the same holds if instead all the frames intersect a vertical line. Chakraborty et al. [8] provided an 8-approximation algorithm on intersection graphs of L-frames intersecting a common vertical line. They also showed that there is a $O(k^4)$-approximation algorithm on unit B_k-VPG graphs[1] and, on the negative side, that the problem is NP-hard on unit B_1-VPG graphs.

To the best of our knowledge, it is not known whether there exist constant-factor approximation algorithms for Independent Set and Dominating Set even on B_1-VPG graphs.

2 Our results

As we have just mentioned, three natural constraints on VPG graphs have been considered in the search for efficient algorithms for Independent Set and Dominating Set: bound the number of bends on each path, bound the number of intersections between any two paths and bound the lengths of segments of paths. Unfortunately, combining these constraints is not enough to guarantee polynomial-time solvability:

▶ **Theorem 2.1.** Independent Set *and* Dominating Set *remain* NP*-complete when restricted to VPG graphs admitting a* 1*-string* B_0*-VPG representation such that each horizontal path has length at most* 2*, even if such representation is part of the input.*

But what about approximation algorithms? Perhaps surprisingly, it turns out that the problems admit PTASes when those constraints are in place. More precisely, we show the following:

▶ **Theorem 2.2.** *Let* $t \geq 0$ *and* $c \geq 1$ *be integers.* Independent Set *and* Dominating Set *admit a PTAS when restricted to VPG graphs with a representation* $\mathcal{R} = (\mathcal{G}, \mathcal{P})$ *such that:*

1. *each path in* $\mathcal{P}$ *has a polynomial (in* $|\mathcal{P}|$*) number of bends;*
2. *each grid-edge in* $\mathcal{G}$ *belongs to at most* t *paths in* $\mathcal{P}$*;*
3. *the horizontal part of each path in* $\mathcal{P}$ *has length at most* c*.*

Here the *horizontal part* of a path is the interval corresponding to the projection of the path onto the horizontal axis. Clearly, for fixed $k \geq 0$, B_k-VPG graphs satisfy condition 1. The class of VPG graphs satisfying condition 2 is rich as well: it contains k-string VPG graphs, for any fixed k, VPG graphs with maximum degree at most $t-1$ and VPG graphs with maximum clique size at most t. Notice that recognizing string graphs (and so VPG graphs) is NP-complete [19, 29]. Similarly, for each fixed $k \geq 0$, recognizing B_k-VPG graphs is NP-complete [9, 21]. Since our PTASes require a VPG representation, we then assume this

[1] A B_k-VPG graph is *unit* if each path consists only of segments with unit length. Notice that every B_k-VPG graph is a unit $B_{k'}$-VPG graph for some finite $k' \geq k$.

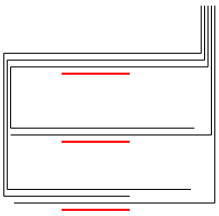


Figure 1: A VPG representation on a bounded number of columns of a split graph whose vertices are partitioned into an independent set I (red paths) and a clique C (black paths).

is always given as part of the input. For each path in the input representation, we maintain a list consisting of its endpoints and bend-points and so the reason behind condition 1 is to avoid the following pathological behavior: there exist string graphs on n vertices requiring paths with $2^{\Omega(n)}$ bends in any representation (this follows from [22]).

▶ Remark. The result for Dominating Set in Theorem 2.2 is best possible in the sense that, if we remove one of conditions 2 and 3, Dominating Set does not admit a PTAS, unless P = NP. Indeed, every split graph admits a VPG representation $\mathcal{R} = (\mathcal{G}, \mathcal{P})$ such that each path in $\mathcal{P}$ has $O(|\mathcal{P}|)$ bends and horizontal part of length at most 3 (see Figure 1). On the other hand, Dominating Set restricted to split graphs cannot be approximated within a factor of $(1 - \varepsilon) \ln n$, for any constant $\varepsilon > 0$, unless NP $\subseteq$ DTIME$(n^{O(\log \log n)})$ [10]. Moreover, every circle graph is a 1-string B_1-VPG graph [1] and Dominating Set is APX-hard on circle graphs [12]. The situation is more subtle for Independent Set, as it has been asked several times whether the problem is APX-hard on B_k-VPG graphs (see, e.g., [5, 25]) and this remains open.

As observed above, the study of these two problems on VPG graphs has focused mostly on B_k-VPG graphs with $k \leq 2$ and, to the best of our knowledge, ours are the first PTASes on a non-trivial subclass of VPG graphs. The PTAS for Dominating Set shows that the constant-factor approximation algorithm on unit B_k-VPG graphs in [8] can in fact be improved if input graphs satisfy also condition 2.

3 Techniques

Our PTASes are obtained by adapting Baker's shifting technique [2] to the string graph setting and showing boundedness of an appropriate width parameter. Baker's technique has already been applied to Independent Set in B_1-EPG graphs [7] and our main contribution is to pair it with powerful mim-width arguments. Our key observation is that if a VPG graph admits a VPG representation with a bounded number of columns and such that each grid-edge belongs to a bounded number of paths, then the graph has bounded mim-width (Theorem 3.3). We then use Baker's shifting technique by solving the problems optimally on each slice with bounded number of columns.

Maximum induced matching width (mim-width for short) is a graph parameter, introduced by Vatshelle [31], measuring how easy it is to decompose a graph along vertex cuts inducing a bipartite graph with small maximum induced matching size. Replacing induced matchings with matchings, one obtains the related parameter called maximum matching width (mm-width for short) [31]. The modelling power of mim-width is stronger than that of treewidth, in the sense that graphs of bounded treewidth have bounded mim-width but there exist graph classes (interval graphs and permutation graphs) with mim-width 1 [31] and unbounded treewidth [15]. On the other hand, mm-width and treewidth are equivalent

parameters, in the sense that one is bounded if and only if the other is [31].

We now properly define mim-width and mm-width. A *branch decomposition* for a graph G is a pair (T, δ), where T is a subcubic tree and δ is a bijection between the vertices of G and the leaves of T. Each edge $e \in E(T)$ naturally splits the leaves of the tree in two groups depending on their component when e is removed. In this way, each edge $e \in E(T)$ represents a partition of $V(G)$ into two partition classes A_e and $\overline{A_e}$, denoted $(A_e, \overline{A_e})$. Denoting by $G[X, Y]$ the bipartite subgraph of G induced by the edges with one endpoint in X and the other in Y, mim-width and mm-width are defined as follows:

▶ **Definition 3.1.** Let G be a graph and let (T, δ) be a branch decomposition for G. For each edge $e \in E(T)$ and the corresponding partition $(A_e, \overline{A_e})$ of $V(G)$, we denote by $\mathrm{cutmim}_G(A_e, \overline{A_e})$ and $\mathrm{cutmm}_G(A_e, \overline{A_e})$ the size of a maximum induced matching and maximum matching in $G[A_e, \overline{A_e}]$, respectively. The mim-width of the branch decomposition (T, δ) is the quantity $\mathrm{mimw}_G(T, \delta) = \max_{e \in E(T)} \mathrm{cutmim}_G(A_e, \overline{A_e})$. The mim-width $\mathrm{mimw}(G)$ of the graph G is the minimum value of $\mathrm{mimw}_G(T, \delta)$ over all possible branch decompositions (T, δ) for G. The mm-width of the branch decomposition (T, δ) is the quantity $\mathrm{mmw}_G(T, \delta) = \max_{e \in E(T)} \mathrm{cutmm}_G(A_e, \overline{A_e})$ and the mm-width $\mathrm{mmw}(G)$ of the graph G is the minimum value of $\mathrm{mmw}_G(T, \delta)$ over all possible branch decompositions (T, δ) for G.

Notice that, for any branch decomposition (T, δ) for G, $\mathrm{mimw}_G(T, \delta) \leq \mathrm{mmw}_G(T, \delta)$ and so $\mathrm{mimw}(G) \leq \mathrm{mmw}(G)$. It is well-known that boundedness of treewidth allows polynomial-time solvability of several otherwise NP-hard graph problems. Boundedness of mim-width has important algorithmic consequences as well, in particular for the so-called (σ, ρ)-domination problems, a subclass of graph problems expressible in MSO_1 introduced by Telle and Proskurowski [30] and including Independent Set and Dominating Set:

▶ **Theorem 3.2** (see, e.g., [17]). *There is an algorithm that, given a graph G and a branch decomposition (T, δ) for G with $w = \mathrm{mimw}_G(T, \delta)$, solves* Independent Set *and* Dominating Set *in $O(n^{4+3w})$ time.*

It should be remarked that, in contrast to treewidth, deciding the mim-width of a graph is W[1]-hard [28]. However, it is possible to find branch decompositions of constant mim-width in polynomial time for several classes of graphs such as permutation graphs, convex graphs [4], H-graphs [13], leaf powers [17].

▶ **Theorem 3.3.** *Let G be a VPG graph with a representation $\mathcal{R} = (\mathcal{G}, \mathcal{P})$ such that each grid-edge in $\mathcal{G}$ belongs to at most t paths in $\mathcal{P}$ and $\mathcal{G}$ contains at most ℓ columns, for some integers $t, \ell \geq 0$. Then $\mathrm{mimw}(G) \leq \mathrm{mmw}(G) \leq 3t \cdot (\ell + 1)$. Moreover, if we are given a VPG graph G on n vertices together with a representation $\mathcal{R} = (\mathcal{G}, \mathcal{P})$ as above and such that in addition each path in $\mathcal{P}$ has a number of bends polynomial in n, then it is possible to compute in $O(n \log n)$ time a branch decomposition (T, δ) for G such that $\mathrm{mimw}_G(T, \delta) \leq \mathrm{mmw}_G(T, \delta) \leq 3t \cdot (\ell + 1)$.*

▶ Remark. Theorem 3.3 is best possible in the following strong sense: both conditions on the VPG graph are necessary to guarantee boundedness of mim-width. As for the first, consider split graphs. They admit a VPG representation $(\mathcal{G}, \mathcal{P})$ such that $\mathcal{G}$ contains at most 4 columns (see Figure 1) but they do not have bounded mim-width [27]. As for the second, consider grid graphs. Since any grid graph is planar and bipartite, it is a contact graph of a family $\mathcal{P}$ of interiorly disjoint segments on a rectangular grid $\mathcal{G}$ [11] and hence admits a VPG representation $(\mathcal{G}, \mathcal{P})$ such that each grid-edge in $\mathcal{G}$ belongs to at most one path in $\mathcal{P}$. On the other hand, grid graphs do not have bounded mim-width [31].

Recalling that for any graph G, $\mathrm{tw}(G) \leq 3 \cdot \mathrm{mmw}(G) - 1$ [18, 31], the graphs satisfying the assumptions in Theorem 3.3 have treewidth at most $9t \cdot (\ell + 1) - 1$. However, in bounding treewidth for these graphs, it seems more natural to work with branch decompositions rather than directly using tree decompositions as in the classical definition of treewidth. We conclude with a sketch of the proof of Theorem 2.2:

Sketch of proof. We provide the PTAS for Dominating Set. Let G be an input graph on n vertices with grid representation $\mathcal{R} = (\mathcal{G}, \mathcal{P})$. We assume that G is connected. This implies that no column in $\mathcal{G}$ is unused and so the number of columns m is at most $(c+1)n$. Without loss of generality, the paths in $\mathcal{P}$ contain only grid-points with x-coordinates between 0 and $m-1$ and y-coordinates at least 0. Given $0 < \varepsilon < 1$, we fix $k = \lceil c(\frac{2}{\varepsilon} - 1) \rceil$. Clearly, $k > c$. We finally assume that $m > k + 2c - 1$, or else we can compute an exact solution by Theorems 3.2 and 3.3. We then proceed as follows. For a fixed $s \in \{0, \ldots, k + c - 1\}$, we let $r = \lceil \frac{m-s-c}{k+c} \rceil$. For any $i \in \{0, \ldots, m-2\}$, we compute in time polynomial in n the set X_i of vertices whose corresponding path contains a grid-edge $[(i, j), (i+1, j)]$ for some $j \in \mathbb{N}$ (we also set $X_{-1} = \varnothing$ and $X_{m-1} = \varnothing$). For any $j \in \{0, \ldots, m-1\}$, we compute in time polynomial in n the set V_j of vertices whose corresponding path intersects column j. We then partition G into slices as follows. For any $i \in \{0, \ldots, r\}$, we define $V(i, s)$ and the set $E(i, s)$ of *exterior vertices* of $V(i, s)$ as follows (see Figure 2):

- $V(0, s) = \bigcup_{\ell=0}^{s+c-1} V_\ell$ and $E(0, s) = X_{s+c-1}$;
- For $0 < i < r$, $V(i, s) = \bigcup_{\ell=0}^{k+2c-1} V_{(i-1)\cdot(k+c)+s+\ell}$ and $E(i, s)$ is the union of $E_L(i, s) = X_{(i-1)\cdot(k+c)+s-1}$ and $E_R(i, s) = X_{i\cdot(k+c)+s+c-1}$;
- $V(r, s) = \bigcup_{\ell=(r-1)\cdot(k+c)+s}^{m-1} V_\ell$ and $E(r, s) = X_{(r-1)\cdot(k+c)+s-1}$.

Moreover, for any $i \in \{0, \ldots, r\}$, the set $I(i, s)$ of *interior vertices* of $V(i, s)$ is defined as $V(i, s) \setminus E(i, s)$. For $i \in \{0, r\}$, we now let $G_I(i, s) = G[I(i, s)]$ and let $G_{LR}(i, s)$ be the graph with vertex set $V(i, s)$ and edge set $E(G[V(i, s)]) \cup \{uv : u, v \in E(i, s)\}$. Moreover, for $0 < i < r$: $G_I(i, s) = G[I(i, s)]$; $G_L(i, s)$ is the graph with vertex set $I_L(i, s) = I(i, s) \cup E_L(i, s)$ and edge set $E(G[I_L(i, s)]) \cup \{uv : u, v \in E_L(i, s)\}$; $G_R(i, s)$ is the graph with vertex set $I_R(i, s) = I(i, s) \cup E_R(i, s)$ and edge set $E(G[I_R(i, s)]) \cup \{uv : u, v \in E_R(i, s)\}$; $G_{LR}(i, s)$ is the graph with vertex set $V(i, s)$ and edge set $E(G[V(i, s)]) \cup \{uv : u, v \in E_L(i, s)\} \cup \{uv : u, v \in E_R(i, s)\}$.

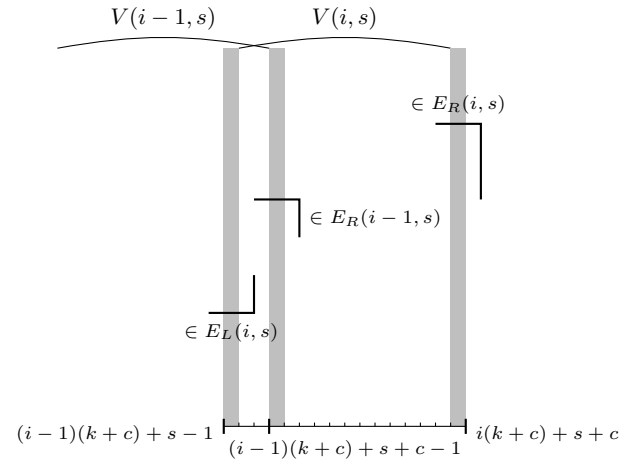


Figure 2: The partition of the input graph G into slices.

We then observe that, for each i such that the corresponding graphs are defined, the VPG representations of $G_I(i,s)$, $G[V(i,s)]$, $G[I_L(i,s)]$ and $G[I_R(i,s)]$ induced by $\mathcal{R}$ contain a bounded number of columns. Using Theorems 3.2 and 3.3, we then compute minimum dominating sets $S_I(i,s)$, $S_L(i,s)$, $S_R(i,s)$ and $S_{LR}(i,s)$ of $G_I(i,s)$, $G_L(i,s)$, $G_R(i,s)$ and $G_{LR}(i,s)$, respectively, in time polynomial in n. Now, for $0 < i < r$, let $S(i,s)$ be a set of minimum cardinality among $S_I(i,s)$, $S_L(i,s)$, $S_R(i,s)$ and $S_{LR}(i,s)$. Similarly, for $i \in \{0,r\}$, let $S(i,s)$ be a set of minimum cardinality among $S_I(i,s)$ and $S_{LR}(i,s)$. We then repeat the procedure above for each fixed $s \in \{0,\ldots,k+c-1\}$ in order to compute the sets $S_s = \bigcup_{i=0}^{r} S(i,s)$ and return the smallest set S among the S_s's in time polynomial in n. It can be shown that S is a dominating set of G such that $|S| \leq (1+\varepsilon)|\mathsf{OPT}|$. ◀

References

1 A. Asinowski, E. Cohen, M. C. Golumbic, V. Limouzy, M. Lipshteyn, and M. Stern. Vertex intersection graphs of paths on a grid. *Journal of Graph Algorithms and Applications*, 16(2):129–150, 2012.

2 B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. ACM*, 41(1):153–180, 1994.

3 S. Bandyapadhyay, A. Maheshwari, S. Mehrabi, and S. Suri. Approximating dominating set on intersection graphs of rectangles and L-frames. *Computational Geometry*, 82:32 44, 2019.

4 R. Belmonte and M. Vatshelle. Graph classes with structured neighborhoods and algorithmic applications. *Theoretical Computer Science*, 511:54–65, 2013.

5 T. Biedl and M. Derka. Splitting B_2-VPG graphs into outer-string and co-comparability graphs. In F. Ellen, A. Kolokolova, and J.-R. Sack, editors, *Algorithms and Data Structures*, pages 157–168. Springer International Publishing, 2017.

6 P. Bose, P. Carmi, J. M. Keil, A. Maheshwari, S. Mehrabi, D. Mondal, and M. Smid. Computing maximum independent set on outerstring graphs and their relatives. In Z. Friggstad, J.-R. Sack, and M. R. Salavatipour, editors, *Algorithms and Data Structures*, pages 211–224. Springer International Publishing, 2019.

7 M. Bougeret, S. Bessy, D. Gonçalves, and C. Paul. On Independent Set on B_1-EPG graphs. In L. Sanità and M. Skutella, editors, *Approximation and Online Algorithms*, pages 158–169. Springer International Publishing, 2015.

8 D. Chakraborty, S. Das, and J. Mukherjee. Approximating Minimum Dominating Set on string graphs. In I. Sau and D. M. Thilikos, editors, *Graph-Theoretic Concepts in Computer Science*, pages 232–243. Springer International Publishing, 2019.

9 S. Chaplick, V. Jelínek, J. Kratochvíl, and T. Vyskočil. Bend-bounded path intersection graphs: Sausages, noodles, and waffles on a grill. In M. C. Golumbic, M. Stern, A. Levy, and G. Morgenstern, editors, *Graph-Theoretic Concepts in Computer Science*, pages 274–285. Springer Berlin Heidelberg, 2012.

10 M. Chlebík and J. Chlebíková. Approximation hardness of dominating set problems in bounded degree graphs. *Information and Computation*, 206(11):1264–1275, 2008.

11 J. Czyzowicz, E. Kranakis, and J. Urrutia. A simple proof of the representation of bipartite planar graphs as the contact graphs of orthogonal straight line segments. *Information Processing Letters*, 66(3):125–126, 1998.

12 M. Damian and S. V. Pemmaraju. APX-hardness of domination problems in circle graphs. *Information Processing Letters*, 97(6):231–237, 2006.

13 F. V. Fomin, P. A. Golovach, and J.-F. Raymond. On the tractability of optimization problems on h-graphs. *Algorithmica*, 82(9):2432–2473, 2020.

14 J. Fox and J. Pach. Computing the independence number of intersection graphs. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '11, pages 1161–1165. Society for Industrial and Applied Mathematics, 2011.

15 M. C. Golumbic and U. Rotics. On the clique-width of some perfect graph classes. *International Journal of Foundations of Computer Science*, 11(03):423–443, 2000.

16 D. Gonçalves, L. Isenmann, and C. Pennarun. Planar graphs as L-intersection or L-contact graphs. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '18, pages 172–184. Society for Industrial and Applied Mathematics, 2018.

17 L. Jaffke, O. Kwon, T. J. F. Strømme, and J. A. Telle. Mim-width III. Graph powers and generalized distance domination problems. *Theoretical Computer Science*, 796:216–236, 2019.

18 J. Jeong, S. H. Sæther, and J. A. Telle. Maximum matching width: New characterizations and a fast algorithm for dominating set. *Discrete Applied Mathematics*, 248:114–124, 2018.

19 J. Kratochvíl. String graphs. II. Recognizing string graphs is NP-hard. *Journal of Combinatorial Theory, Series B*, 52(1):67–78, 1991.

20 J. Kratochvíl. String graphs. I. The number of critical nonstring graphs is infinite. *Journal of Combinatorial Theory, Series B*, 52(1):53–66, 1991.

21 J. Kratochvíl. A special planar satisfiability problem and a consequence of its NP-completeness. *Discrete Applied Mathematics*, 52(3):233–252, 1994.

22 J. Kratochvíl and J. Matoušek. String graphs requiring exponential representations. *Journal of Combinatorial Theory, Series B*, 53(1):1–4, 1991.

23 J. Kratochvíl and J. Nešetřil. INDEPENDENT SET and CLIQUE problems in intersection-defined classes of graphs. *Commentationes Mathematicae Universitatis Carolinae*, 31(1):85–93, 1990.

24 A. Lahiri, J. Mukherjee, and C. R. Subramanian. Maximum Independent Set on B_1-VPG graphs. In Z. Lu, D. Kim, W. Wu, W. Li, and D.-Z. Du, editors, *Combinatorial Optimization and Applications*, pages 633–646. Springer International Publishing, 2015.

25 S. Mehrabi. Approximation algorithms for independence and domination on B_1-VPG and B_1-EPG graphs. *CoRR*, abs/1702.05633, 2017. URL `https://arxiv.org/abs/1702.05633`.

26 S. Mehrabi. Approximating domination on intersection graphs of paths on a grid. In R. Solis-Oba and R. Fleischer, editors, *Approximation and Online Algorithms*, pages 76–89. Springer International Publishing, 2018.

27 S. Mengel. Lower bounds on the mim-width of some graph classes. *Discrete Applied Mathematics*, 248:28–32, 2018.

28 S. H. Sæther and M. Vatshelle. Hardness of computing width parameters based on branch decompositions over the vertex set. *Theoretical Computer Science*, 615:120–125, 2016.

29 M. Schaefer, E. Sedgwick, and Daniel Štefankovič. Recognizing string graphs in NP. *Journal of Computer and System Sciences*, 67(2):365–380, 2003.

30 J. A. Telle and A. Proskurowski. Algorithms for vertex partitioning problems on partial k-trees. *SIAM Journal on Discrete Mathematics*, 10(4):529–550, 1997.

31 M. Vatshelle. *New Width Parameters of Graphs*. PhD thesis, University of Bergen, 2012.

Covering a Curve with Subtrajectories*

Hugo A. Akitaya[1], Frederik Brüning[2], Erin Chambers[3], and Anne Driemel[4]

1 **School of Computer Science, Carleton University, Canada**
2 **Department of Computer Science, University of Bonn, Germany**
3 **Department of Computer Science, Saint Louis University, USA**
4 **Hausdorff Center for Mathematics, University of Bonn, Germany**

Abstract

We study subtrajectory clustering under the Fréchet distance. Given a polygonal curve P with n vertices, and parameters k and ℓ, the goal is to find k center curves of complexity at most ℓ such that every point on P is covered by a subtrajectory that has small Fréchet distance to one of the k center curves. We suggest a new approach to solving this problem based on a set cover formulation leading to polynomial-time approximation algorithms. Our solutions rely on carefully designed set system oracles for systems of subtrajectories.

1 Introduction

Many applications use recorded sequences of positions of moving objects, such as migrating animals, sports players, traffic. Other types of movement are also possible, such as in gesture analysis or eye tracking. Given the nature of the data aquisition, we usually model such trajectory data as piecewise linear curves in $\mathbb{R}^d$. One particular question which has gotten much attention relates to clustering trajectory data; typically, one wishes to extract good representative curves that summarize the data well. This necessitates a notion of similarity to compare and evaluate simplified representations of curves. The Fréchet distance is one such measure, which in addition to geometric closeness also takes the flow of the curve into account; see the next section for the precise definition. In this paper, we consider the problem of covering an input curve with a small set of representative curves, where every portion of the input curve is similar to some representative under the Fréchet distance. Our main focus is to recast this problem in a set system framework and to study approximation algorithms for the corresponding set cover problem.

2 Related work

There are a number of heuristic approaches for clustering trajectories [19] as well as more theoretical approaches based on similarity measures [18] and relative homology [17]. The problem is often studied in a setting where each of the trajectories to be clustered is given as an immutable element of a specific metric space.

In subtrajectory clustering, however, the goal is to find self-similarities within one long trajectory (see Figure 1 for an example). To this end, we ask to find a suitable decomposition that splits the input trajectory into smaller subsections (subtrajectories) that can be clustered.

* This research was initiated at the Eighth Annual Workshop on Geometry and Graphs, held at the Bellairs Research Institute in Barbados, January 31 – February 7, 2020. The authors are grateful to the organizers and to the participants of this workshop. Anne Driemel acknowledges funding from the DFG (project nr. 313421352). Hugo Akitaya is supported by NSERC. Erin Chambers acknowledges funding from the National Science Foundation through grants CCF-1614562, CCF-1907612 and DBI-1759807. A full version of this article is available on arXiv [2].

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

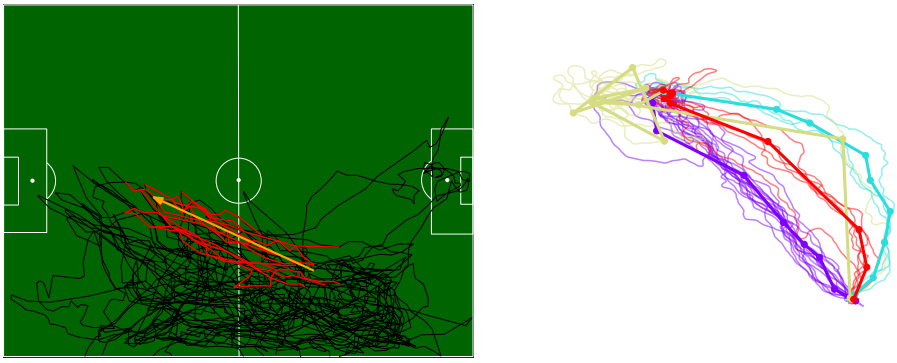


Figure 1 Left: Example of a possible subtrajectory cluster within the trajectory of a soccer player; Right: Example (k, ℓ)-clustering [8].

Alternatively, we may ask for a covering instead of a decomposition by allowing subtrajectories to overlap. There are many different variants of subtrajectory clustering studied and heuristics for finding subtrajectory clusters have been proposed [14, 15]. One of the earlier theoretical results for subtrajectory clustering under the Fréchet distance [6, 13] focuses on finding one cluster of long subtrajectories that are similar to each other, optimizing different parameters. They also show that finding an optimal cluster is NP-complete. In subsequent work, the algorithmic techniques were shown to be useful for map construction [4, 5]. Recently, the authors of [1] proposed to study facility location for subtrajectory clustering under the discrete Fréchet distance. They present $O(\log m)$-approximation algorithms, where m is the number of points on the input curve. The main algorithm runs in $\tilde{O}(|B|m^3)$ if B is a set of candidate center curves given with the input. They show how to generate a suitable set B of size $O(m^2)$, and how to reduce the size to $O(m)$ at the expense of an additional $O(\log m)$-factor in the approximation quality. Our work is different from [1] in many ways. We restrict the complexity of admissible center curves and we use the continuous Fréchet distance. The authors of [1] also consider a set cover problem as an intermediate step of their algorithm, but their formulation leads to a set system of exponential size. Our approach instead puts a set system at the center of the problem definition, and our main contribution is a careful formulation of suitable set systems of much smaller size that lend themselves to efficient approximation. Our work draws from ideas and techniques developed in works on the (k, ℓ)-clustering variant for trajectories [12, 7, 16, 9] (see Figure 1 for an exmaple), as well as fundamental work on computing hitting sets of set systems for low VC-dimension [3], which have previously not been applied to subtrajectory clustering. Given a polygonal curve P with n vertices, and parameters k and ℓ, the goal is to find k center curves of complexity at most ℓ such that every point on P is covered by a subtrajectory, that has small Fréchet distance to one of the k center curves.

3 Preliminaries

A sequence of n points $p_1, \ldots, p_n \in \mathbb{R}^d$ defines a ***polygonal curve*** P by linearly interpolating consecutive points, that is, for each i, we obtain the ***edge*** $\{tp_i + (1-t)p_{i+1} | t \in [0, 1]\}$. We may think of P resulting from the concatenation of the edges in the given order as a parametrized curve, that is, a function $P : [0, 1] \mapsto \mathbb{R}^d$. Note that for any such parametrized curve we can find real values $s_1 \leq \cdots \leq s_n$, such that $P(s_i) = p_i$. We call the ordered set of the p_i the ***vertices*** of P and we denote it with $V(P)$. We call the number of vertices n the

complexity of the curve. For any two $a, b \in [0,1]$ we denote with $P[a,b]$ the ***subcurve*** of P that starts at $P(a)$ and ends at $P(b)$. Let $\mathbb{X}_\ell^d = (\mathbb{R}^d)^\ell$. We think of the elements of this set as sequences of ℓ points in $\mathbb{R}^d$, the set of all polygonal curves of ℓ vertices in $\mathbb{R}^d$.

For two parametrized curves P and Q, we define their ***Fréchet distance*** as

$$d_F(P,Q) = \inf_{\gamma:[0,1]\mapsto[0,1]} \sup_{t\in[0,1]} \|P(\gamma(t)) - Q(t)\|,$$

where γ ranges over all strictly monotone increasing functions. A curve $Q \in \mathbb{X}_\ell^d$ is an ℓ-***simplification*** of a curve P if its Fréchet distance is minimum among all curves in $\mathbb{X}_\ell^d$. Let X be a set. We call a set $\mathcal{R}$, where any $r \in \mathcal{R}$ is of the form $r \subseteq X$ a ***set system*** with ***ground set*** X. Let $\mathcal{R}$ be a set system with ground set X. A ***set cover*** of $\mathcal{R}$ is a subset $S \subset \mathcal{R}$ such that the ground set is covered by the union of the sets in S. The ***set cover problem*** asks to find a set cover for a given $\mathcal{R}$ of minimum size.

4 Problem definition

Let P be a polygonal curve given by a sequence of points $p_1, \ldots, p_n \in \mathbb{R}^d$ and endowed with a set of m real values $0 = t_1 < t_2 < \cdots < t_m = 1$ which define a set of subcurves of the form $S_{ij} = P[t_i, t_j]$. Consider set of values t_i, we refer to the respective points on the curve $P(t_i)$ for $1 \le i \le m$ as ***breakpoints***. Let $\ell \in \mathbb{N}$ and $\Delta \in \mathbb{R}$ be fixed parameters.

Consider the set system $\mathcal{R}$ with ground set $Z = \{1, \ldots, m-1\}$ where each set $r_Q \in \mathcal{R}$ is defined by a polygonal curve $Q \in \mathbb{X}_\ell^d$ as follows

$$r_Q = \{z \in Z \mid \exists i \le z < j \text{ with } d_F(Q, P[t_i, t_j]) \le \Delta\} \tag{1}$$

The main problem we are studying in this paper is to compute minimum-size set covers for set systems defined as above; the set system framework allows us to draw from a rich background of techniques in computing set covers. Our motivation is the related clustering problem, where we think of Q as a candidate for a cluster center and the curves $P[t_i, t_j]$ as subcurves that are to be clustered. The goal is to cover the entire curve P with a small set of metric balls centered at polygonal curves of low complexity. More precisely, we want to find a minimum-size set of polygonal curves $C \subset \mathbb{X}_\ell^d$, such that the cost of clustering P under the Fréchet distance is at most Δ; here, we break P into subcurves determined by the breakpoints, each of which must match to some curve of C. We define the cost function

$$\phi(P, C) = \min_{I \subseteq \mathcal{S}} \max_{(i,j)\in I} \min_{q \in C} d_F(P[t_i, t_j], q),$$

where $\mathcal{S} = \{(i,j) \in \mathbb{N}^2 \mid 1 \le i < j \le m\}$, and we require that I satisfies

$$[0,1] = \bigcup_{(i,j)\in I} [t_i, t_j].$$

I represents a set of intervals starting and ending at breakpoints that together cover the parametrization interval of P. The problem we study in this paper is to find a set of cluster centers $C \subset \mathbb{X}_\ell^d$ of size k, such that $\phi(P, C) \le \Delta$. It is important to note that we do not compute a clustering of the set of subcurves of P. Instead, we want to find a covering of the curve P with subcurves that allows for a good clustering.

5 Results

Using the techniques described further below, we can show results of the following form.

Let $P : [0,1] \to \mathbb{R}^d$ be a polygonal curve of complexity n with breakpoints $0 \leq t_1, \ldots, t_m \leq 1$. Assume there exists a set $C \subset \mathbb{X}_\ell^d$ of size k, such that $\phi(P, C) \leq \Delta$. We prove that there exists an algorithm that computes an approximation to the exact solution: a set $C' \subset \mathbb{X}_{\ell'}^d$ of size k' such that $\phi(P, C') \leq \Delta'$, for several possible target approximations k', ℓ', and Δ' that depend upon k, ℓ, and Δ. Table 1 gives an overview of our running times and approximation quality. Details are in the full version [2], and below we sketch the main ideas. In addition, we show in the full version that, even when fixing $\ell = 1$ and minimizing k, this problem is NP-Hard, via a reduction from Planar-Monotone-3SAT.

k'	ℓ'	Δ'	running time
$O(k \log m)$	ℓ	3Δ	$O(m^3 n\ell + m^4 + m^2 T_S(n, \ell)))$
$O(k \log m)$	ℓ	$O(\Delta)$	$\widetilde{O}(m^4\ell^2 + n + m^2 T_S(n, \ell))$
$O(k \log k \log m \operatorname{loglog} m)$	2ℓ	$O(\Delta)$	$\widetilde{O}\left(m^2 k^2 \ell^2 + mn\right)$

Table 1 Overview of results. We use $\widetilde{O}(\cdot)$ to denote asymptotic running times omitting logarithmic factors. $T_S(n, \ell)$ is the running time to compute an ℓ-simplification of a curve of $P[t_i, t_j]$ for any $0 \leq i < j \leq 1$.

6 A set system for approximation

We modify the set system in a way that preserves the initial structure but allows for more efficient approximations of the clustering problem. Starting from the center curve problem [7], let $\mathcal{P} \subseteq \mathbb{X}_N^d$ be a set of curves. Assume $\ell \in \mathbb{N}$ is fixed. For $q \in \mathcal{P}$ let $\mu_\ell(q)$ be a ℓ-simplification of q. By the triangle inequality, it holds for any $q \in \mathcal{P}$, and for any $c \in \mathbb{X}_\ell^d$,

$$\max_{p \in \mathcal{P}} d_F(p, \mu_\ell(q)) \leq \max_{p \in \mathcal{P}} \left(d_F(p, c) + d_F(c, q) + d_F(q, \mu_\ell(q))\right) \leq 3 \max_{p \in \mathcal{P}} d_F(p, c)$$

In particular, this holds for c being the optimal 1-center curve which minimizes the maximum Fréchet distance to the curves in $\mathcal{P}$. At the same time, the simplification $\mu_\ell(q)$ is a valid solution to the center curve problem. This implies that $\mu_\ell(q)$ is a 3-approximation to the optimal 1-center. Therefore, this yields a 3-approximation to the center curve problem, as long as an optimal simplification can be computed exactly.

Consider a set system $\widetilde{\mathcal{R}}_0$ defined on the ground set $Z = \{1, \ldots, m-1\}$, where each set $r_{i,j} \in \widetilde{\mathcal{R}}_0$ is of the form

$$r_{i,j} = \{z \in Z \mid \exists i' \leq z < j' \text{ with } d_F(P[t_{i'}, t_{j'}], \mu_\ell(P[t_i, t_j])) \leq 3\Delta\}.$$

▶ **Lemma 1.** *For any $r_Q \in \mathcal{R}$, there is a $r_{i,j} \in \widetilde{\mathcal{R}}_0$ such that $r_Q \subseteq r_{i,j}$.*

Proof. We can rewrite the definition of r_Q as follows. Let Y be the set of tuples $(i, j) \in \mathbb{N}^2$ with $1 \leq i < j \leq m$ and $d_F(Q, P[t_i, t_j]) \leq \Delta$. We have that $r_Q = \bigcup_{(i,j) \in Y} [i, j) \cap \mathbb{N}$. Let $(i, j), (i', j') \in Y$. Using the triangle inequality, we can bound $d_F(P[t_{i'}, t_{j'}], \mu_\ell(P[t_i, t_j]))$ by

$$d_F(P[t_{i'}, t_{j'}], Q) + d_F(Q, P[t_i, t_j]) + d_F(P[t_i, t_j], \mu_\ell(P[t_i, t_j])) \leq 3\Delta.$$

By the definition of $r_{i,j}$, we have $[i', j') \cap \mathbb{N} \subseteq r_{i,j}$ and therefore $r_Q \subseteq r_{i,j}$. In other words, we can choose any maximal set of covered intervals within r_Q and use the simplification of the corresponding subcurve of P to find a set of $\widetilde{\mathcal{R}}_0$ that includes r_Q. ◀

The well-known greedy set cover algorithm yields an $O(\log(m))$ approximation for a ground set of size m [10]. The algorithm works as follows: It incrementally builds a set cover by taking the set with the largest number of uncovered elements in each step. We apply this algorithm to the set system $\widetilde{\mathcal{R}}_0$. From the resulting set cover we obtain a set C consisting of ℓ-simplifications $\mu_\ell(P[t_i, t_j]))$ for each $r_{i,j}$, such that $\phi(P, C) \leq 3\Delta$. Then, it remains to show that efficient set system oracles can be built for $\widetilde{\mathcal{R}}_0$. This can be done by employing data structures for Fréchet queries, such as the one by Driemel and Har-Peled [11]. From this we obtain the first two results in Table 1.

7 A faster algorithm

Our main result is an adaptation of the set cover algorithm by Brönniman and Goodrich [3] that improves upon the greedy set cover in terms of running time. The result is stated in the bottom line of Table 1. A crucial element is the formulation of a suitable set system for approximation. Here, we describe such a set system for $\ell = 2$. We define the set system $\widetilde{\mathcal{R}}_2$ with ground set $Z = \{1, \ldots, m-1\}$. Denote $\tau_{i,j} = \overline{P(t_i)P(t_j)}$. For a subsequence $S = s_1, \ldots, s_r$ of $1, \ldots, m$, denote

$$\pi(S) = \tau_{s_1,s_2} \oplus \tau_{s_2,s_3} \oplus \cdots \oplus \tau_{s_{r-1},s_r}.$$

A tuple (i, j) with $1 \leq i \leq j \leq m$ defines a set $r_{i,j} \in \widetilde{\mathcal{R}}_2$ as follows

$$r_{i,j} = \{z \in Z \mid \exists x \in [x_z, z], y \in [z+1, y_z] \text{ with } d_F(\pi(x, z, z+1, y), \tau_{i,j})) \leq 2\Delta\},$$

where $x_z \leq z < y_z$ are indices which we obtain as follows. We scan breakpoints starting from z in the backwards order along the curve and to test for each breakpoint x, whether

$$d_F(\tau_{x,z}, P[t_x, t_z]) \leq 4\Delta. \tag{2}$$

If x satisfies (2), then we decrement x and continue the scan. If $x = 0$ or if x does not satisfy (2), then we set $x_z = x + 1$ and stop the scan. To set y_z we use a similar approach: We scan forwards from $z+1$ along the curve and test for each breakpoint y the same property with $\tau_{z+1,y}$ and $P[t_{z+1}, t_y]$. If y satisfies the property, we increment y and continue the scan. If $y = m+1$ or if y does not satisfy the property we set $y_z = y-1$ and stop the scan. Figure 2 shows an example of z, x_z and y_z, and Figure 3 illustrates the idea of the query algorithm.

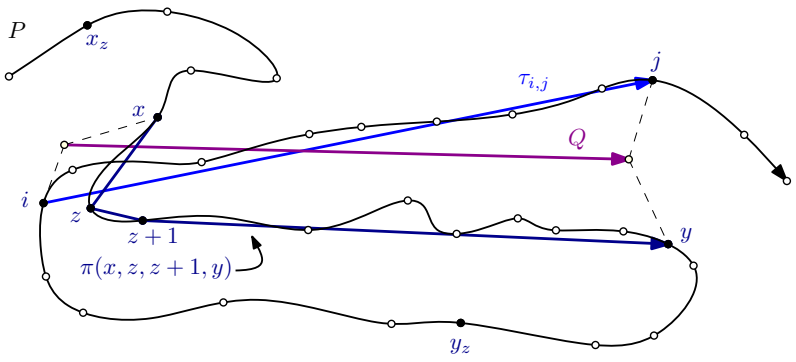


Figure 2 Example of a curve P and index z, such that $z \in r_{i,j}$ for some $r_{i,j} \in \widetilde{\mathcal{R}}_2$. Also shown is a line segment Q, such that $z \in r_Q$ of the initial set system $\mathcal{R}$. After preprocessing, we can test $z \in r_{i,j}$ in constant time while using space in $O(m^2 + n)$.

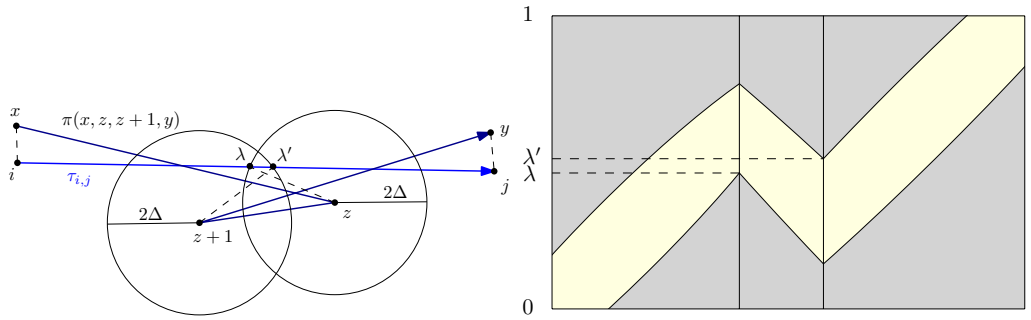


Figure 3 Illustration of the query $z \in r_{i,j}$. The query checks if a monotone path is feasible in the 2Δ-free-space diagram (right) of a subcurve of the proxy curve $\pi(x, z, z+1, y)$ of z and a query $\tau_{i,j}$ (left) for any $x \in [x_z, z]$ and $y \in [z+1, y_z]$. We can show that the query $z \in r_{i,j}$ can be answered in $O(1)$ time (after preprocessing).

7.1 The set system oracle

We describe a set system oracle for $\widetilde{\mathcal{R}}_2$. In particular, we show how to build a data structure that answers a query, given indices i, j and z, for the predicate $z \in r_{i,j}$ in $O(1)$ time.

To build the data structure for the oracle, we first compute the indices x_z and y_z for each $1 \leq z \leq m-1$, as specified in the definition of the set system. Next, we construct a data structure that can answer for a pair of breakpoints i and z if there is a breakpoint x with $x_z \leq x \leq z$ such that $\|P(t_x) - P(t_i)\| \leq 2\Delta$ in $O(1)$ time. For this we build an $m \times m$ matrix M in the following way. For each breakpoint i we go through the sorted list of breakpoints and check if $\|P(t_i) - P(t_j)\| \leq 2\Delta$ for each $1 \leq j \leq m$. While doing that, we determine for each j which is the first breakpoint $z_{i,j} \geq j$ with $\|P(t_i) - P(t_{z_{i,j}})\| \leq 2\Delta$. The entries $z_{i,j}$ are then stored in the matrix M at position $M(i, j)$. Given the Matrix M the oracle can answer if there is a breakpoint x with $x_z \leq x \leq z$ such that $\|P(t_x) - P(t_i)\| \leq 2\Delta$ by checking if $M(i, x_z) \leq z$. The data structure can also answer if there is a breakpoint y with $z+1 \leq y \leq y_z$ such that $\|P(t_y) - P(t_j)\| \leq 2\Delta$ by checking if $M(j, z+1) \leq y_z$. The final data structure stores the matrix M only.

We answer queries as follows. Given z, i and j, we want to determine if $z \in r_{i,j}$. We return "yes", if the following three conditions are satisfied: (i) $M(i, x_z) \leq z$ (ii) $M(j, z+1) \leq y_z$ (iii) $\|s - P(t_z)\| \leq 2\Delta$, where s is the intersection of the bisector between the points $P(t_z)$ and $P(t_{z+1})$ and the line segment $\tau_{i,j}$. Otherwise, the algorithm returns "no".

Correctness is implied by the following lemma.

▶ **Lemma 2.** *$d_F(\pi(x, z, z+1, y), \tau_{i,j}) \leq 2\Delta$ if and only if the following three conditions are satisfied: (i) $\|P(t_x) - u\| \leq 2\Delta$ (ii) $\|P(t_y) - v\| \leq 2\Delta$ (iii) $\min_{\substack{\lambda, \lambda' \in [0,1] \\ \lambda \leq \lambda'}} (\|a - (\lambda v + (1 - \lambda)u)\|, \|b - (\lambda' v + (1-\lambda')u)\|) \leq 2\Delta$ where $a = P(t_z)$, $b = P(t_{z+1})$, $u = P(t_i)$, and $v = P(t_j)$.*

7.2 The general case

For the general case, where $\ell \geq 2$, we replace the edges of the proxy curve π in the above set system by simplifications of the corresponding subcurves. We show that it is possible to ensure that these simplifications are nested in a certain way. This in turn allows us to build efficient data structures for this set system, leading to the third result in Table 1.

References

1. Pankaj K. Agarwal, Kyle Fox, Kamesh Munagala, Abhinandan Nath, Jiangwei Pan, and Erin Taylor. Subtrajectory clustering: Models and algorithms. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, SIGMOD/PODS '18, page 75–87, New York, NY, USA, 2018. Association for Computing Machinery.
2. Hugo A. Akitaya, Frederik Brüning, Erin Chambers, and Anne Driemel. Covering a curve with subtrajectories. *CoRR*, arXiv:2103.06040, 2021.
3. Hervé Brönnimann and Michael T Goodrich. Almost optimal set covers in finite vc-dimension. *Discrete & Computational Geometry*, 14(4):463–479, 1995.
4. Kevin Buchin, Maike Buchin, David Duran, Brittany Terese Fasy, Roel Jacobs, Vera Sacristan, Rodrigo I. Silveira, Frank Staals, and Carola Wenk. Clustering trajectories for map construction. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '17, New York, NY, USA, 2017. Association for Computing Machinery.
5. Kevin Buchin, Maike Buchin, Joachim Gudmundsson, Jorren Hendriks, Erfan Hosseini Sereshgi, Vera Sacristán, Rodrigo I. Silveira, Jorrick Sleijster, Frank Staals, and Carola Wenk. Improved map construction using subtrajectory clustering. In *Proceedings of the 4th ACM SIGSPATIAL Workshop on Location-Based Recommendations, Geosocial Networks, and Geoadvertising*, LocalRec'20, New York, NY, USA, 2020. Association for Computing Machinery.
6. Kevin Buchin, Maike Buchin, Joachim Gudmundsson, Maarten Löffler, and Jun Luo. Detecting commuting patterns by clustering subtrajectories. In Seok-Hee Hong, Hiroshi Nagamochi, and Takuro Fukunaga, editors, *Algorithms and Computation*, pages 644–655, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.
7. Kevin Buchin, Anne Driemel, Joachim Gudmundsson, Michael Horton, Irina Kostitsyna, Maarten Löffler, and Martijn Struijs. Approximating (k, l)-center clustering for curves. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2922–2938. SIAM, 2019.
8. Kevin Buchin, Anne Driemel, Natasja van de L'Isle, and André Nusser. Klcluster: Center-based clustering of trajectories. In *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '19, page 496–499, New York, NY, USA, 2019. Association for Computing Machinery.
9. Maike Buchin, Anne Driemel, and Dennis Rohde. Approximating (k, l)-median clustering for polygonal curves. In *SODA 2021*. SIAM, 2021. (To appear).
10. V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979.
11. Anne Driemel and Sariel Har-Peled. Jaywalking your dog: Computing the Fréchet distance with shortcuts. *SIAM J. Comput.*, 42(5):1830–1866, 2013.
12. Anne Driemel, Amer Krivosija, and Christian Sohler. Clustering time series under the Fréchet distance. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 766–785, 2016.
13. J. Gudmundsson and N. Valladares. A GPU approach to subtrajectory clustering using the Fréchet distance. *IEEE Transactions on Parallel and Distributed Systems*, 26(4):924–937, April 2015.
14. Joachim Gudmundsson, Andreas Thom, and Jan Vahrenhold. Of motifs and goals: Mining trajectory data. In *Proceedings of the 20th International Conference on Advances in*

Geographic Information Systems, SIGSPATIAL '12, page 129–138, New York, NY, USA, 2012. Association for Computing Machinery.

15 Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. Trajectory clustering: a partition-and-group framework. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Beijing, China, June 12-14, 2007*, pages 593–604, 2007.

16 Abhinandan Nath and Erin Taylor. k-median clustering under discrete Fréchet and Hausdorff distances. In *36th International Symposium on Computational Geometry (SoCG 2020)*, volume 164, page 58, 2020.

17 F. T. Pokorny, K. Goldberg, and D. Kragic. Topological trajectory clustering with relative persistent homology. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 16–23, 2016.

18 C. Sung, D. Feldman, and D. Rus. Trajectory clustering for motion prediction. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1547–1552, Oct 2012.

19 Guan Yuan, Penghui Sun, Jie Zhao, Daxing Li, and Canwei Wang. A review of moving object trajectory clustering algorithms. *Artificial Intelligence Review*, 47(1):123–144, March 2016.

Radial Level Planarity with Fixed Embedding*

Guido Brückner[1] and Ignaz Rutter[2]

1 **Karlsruhe Institute of Technology**
brueckner@kit.edu

2 **University of Passau**
rutter@fim.uni-passau.de

Abstract

We study level planarity testing of graphs with a fixed combinatorial embedding for three different notions of combinatorial embeddings, namely the *level embedding*, the *upward embedding* and the *planar embedding*, which allow for increasing degrees of freedom in the corresponding drawings.

For the fixed level embedding there are known, easy to test level planarity criteria. We use these to prove an untangling lemma that plays a key role in a simple level planarity test with fixed upward embedding. This test is then adapted to the case where only the planar embedding is fixed. Further, we characterize radial upward planar embeddings, which lets us extend our results to radial level planarity. No algorithms were previously known for these problems.

1 Introduction

Level planarity and upward planarity are planarity variants for directed graphs that enrich the notion of planarity by imposing additional requirements based on the directions of the edges. A drawing of a directed graph is upward if all edges are drawn as y-monotone curves. In the level planar setting the y-coordinates of the vertices are fixed. Both planarity variants also exist in a radial form, which we view as upward drawings on the surface of an infinite standing cylinder.

There is a natural definition of *topological equivalence* for such drawings. Two planar drawings Γ_1, Γ_2 of a graph G are *equivalent* if there exists an isotopy between them, i.e., they can be continuously transformed into each other such that each intermediate image is planar. For upward planar (level planar) drawings it is required that the intermediate drawings are also upward planar (level planar). Equivalence classes of such drawings can be combinatorially described by so-called *embeddings* (which give the circular order of edges around each vertex), *upward embeddings* (which give the linear orders of incoming and outgoing edges around each vertex), and *level embeddings* (specifying for each level the order of its vertices and the edges that cross it), respectively. These notions also apply to the radial setting, except that there level embeddings specify circular orderings rather than linear ones.

Not every embedding (upward embedding, level embedding) corresponds to a planar drawing (upward planar drawing, level planar drawing). If such a drawing exists, we call the embedding a *planar embedding* (*upward planar embedding*, *level planar embedding*). Given a fixed combinatorial embedding of one of the above types, it therefore makes sense to ask for which planarity notions there exists a planar drawing that induces the given embedding. In this way, combinations of a planarity variant and combinatorial embedding type give instantiations of the planarity testing with fixed embedding problem; see Table 1 for an overview.

* Work partially supported by DFG grant Ru-1903/3-1.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

	type of fixed embedding			
planarity notion	none	embedding	upward	level
planar	$O(n)$ [14]	$O(n)$ (Euler)	n.a.	n.a.
upward	NPC [11]	$O(n^{3/2})$ [10, 5]	$O(n)$ [5]	n.a.
radial upward	NPC [13]	$O(n \log^3 n)$ Thm. 3.2	$O(n)$ Lem. 3.1	n.a.
level	$O(n)$ [15]	$O(n)$ Thm. 4.6	$O(n)$ Thm. 4.4	$O(n)$ Lem. 4.1
radial level	$O(n)$ [4]	$O(n)$ Thm. 4.6	$O(n)$ Thm. 4.4	$O(n)$ Lem. 4.1

Table 1 Overview of known results and our contribution (blue cells).

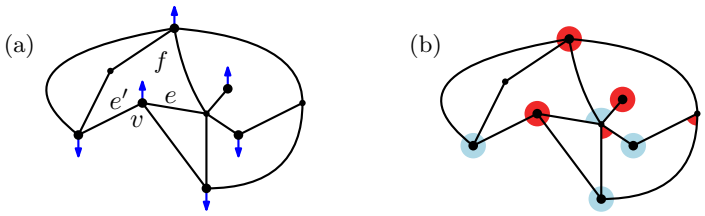


Figure 1 Embedding with a source/sink assignment (a) and face sources/sinks in blue/red (b).

2 Preliminaries

We assume familiarity with basic graph-theoretic concepts. A planar embedding defines faces bounded by *facial walks* $v_1, v_2, \ldots, v_n, v_{n+1} = v_1$ where for each i with $1 \leq i \leq n$ the edge $\{v_{i-1}, v_i\}$ immediately precedes the edge $\{v_i, v_{i+1}\}$ in the counter-clockwise cyclic order of edges incident to v_i. Let $G = (V, E)$ be a directed graph with a planar embedding; see Fig. 1 for an illustration. A vertex v of G is a *source* (*sink*) if all edges incident to v originate from (end at) v. A *source/sink assignment* ψ of G assigns to each source or sink v of G an ordered pair (e, e') of edges where e immediately precedes e' in the counter-clockwise order of edges incident to v [5]. There exists a unique face f on whose facial walk e immediately precedes e', we also say that ψ assigns v to f; see Fig. 1a. An upward planar embedding induces a source/sink assignment by assigning each source v to (e, e'), where e and e' are the leftmost and rightmost outgoing edge incident to v, respectively, and by assigning each sink v to (e, e'), where e and e' are the rightmost and leftmost outgoing edge incident to xv, respectively. Conversely, a planar embedding together with a source/sink assignment induces an upward embedding. We therefore use these concepts interchangeably. Let f be a face and let $v_1, v_2, \ldots, v_n, v_{n+1} = v_1$ denote the facial walk that bounds f. For $1 \leq i \leq n$ the vertex v_i is a *face source* of f if both edges $\{v_{i-1}, v_i\}$ and $\{v_i, v_{i+1}\}$ originate from v_i. Symmetrically, v_i is a *face sink* of f if both edges $\{v_{i-1}, v_i\}$ and $\{v_i, v_{i+1}\}$ have v_i as their endpoint; see Fig. 1b. For each face f let n_f denote the number of face sources of f (cutvertices of G may be counted multiple times).

▶ **Lemma 2.1** ([5],[7, Lemma 1]). *Let G be a connected directed acyclic graph. An upward embedding of G with source/sink assignment ψ is upward planar if and only if there exists a face h (namely, the outer face) such that ψ assigns to each (inner) face $f \neq h$ exactly $n_f - 1$*

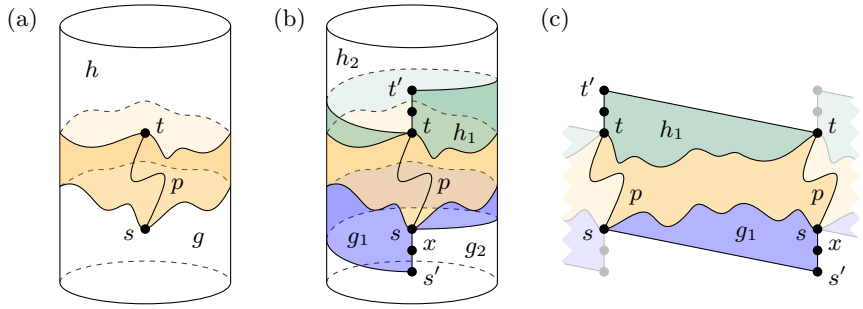


Figure 2 Proof of Lem. 3.1. Extend the radial upward drawing Γ (a) to a radial upward drawing Γ' (b), then unwrap it from the cylinder into the plane by cutting along the path p (c).

sources/sinks, and to h exactly $n_h + 1$ sources/sinks.

When the planar embedding of G is fixed, Bertolazzi et al. provide a quadratic-time algorithm to test whether a source/sink assignment subject to the conditions stated in Lem. 2.1 exists and if so, compute one. This can be improved to $O(n^{3/2})$ [10]. A *level graph* is a directed graph $G = (V, E)$ together with a *level assignment* $\ell : V \to \mathbb{N}$. A *level drawing* is an upward drawing of G where each vertex v has y-coordinate $\ell(v)$. Each level graph G has a *proper form* in which for each edge $(u, v) \in E$ it is $\ell(u) + 1 = \ell(v)$ and whose level embeddings correspond bijectively to the level embeddings of G. It also has a *star form* in which every level hosts exactly one vertex and whose upward planar embeddings correspond bijectively to the upward planar embeddings of G [9, 8]. The following is common knowledge.

▶ **Lemma 2.2.** *A radial level embedding of a level graph G in proper form is radial level planar if for all choices of three independent edges $(u, u'), (v, v'), (w, w')$ of G with $\ell(u) = \ell(v) = \ell(w)$ where u, v, w appear in that cyclic order the vertices u', v', w' appear in that cyclic order.*

3 Radial Upward Planarity

We generalize the upward planarity criterion stated in Lem. 2.1 to radial upward planarity.

▶ **Lemma 3.1.** *Let G be a connected directed acyclic graph. An upward embedding of G is radial upward planar if and only if there exist two faces g, h such that the source/sink assignment ψ assigns to each face f exactly $n_f - 1$ sources/sinks, plus one source if $f = g$, and plus one sink if $f = h$.*

Proof Sketch. "⇒": Let Γ be a radial upward planar drawing G; see Fig. 2(a). Augment Γ as illustrated in Fig. 2 (b). Then cut the drawing at a path p from the new source s' to the new sink t' and unwrap it from the cylinder into the plane; see Fig. 2 (c). Apply Lem. 2.1 on the unwrapped drawing to show the claimed properties. "⇐": Use the source/sink assignment ψ and a result of Bertolazzi et al. (see the if-part of the proof of Thm. 3 in [5]) to augment G to a planar acyclic graph with a single source and a single sink. Such graphs are radial upward planar [12, 3]. ◀

A given upward embedding of G can be tested for radial upward planarity in linear time using the criterion provided by Lem. 3.1. For a planar embedding, the criterion from Lem. 3.1 can be formulated as a maximum flow problem in a 2-apex graph, which can then be solved in $O(n \log^3 n)$ time [6, Cor. 5.1].

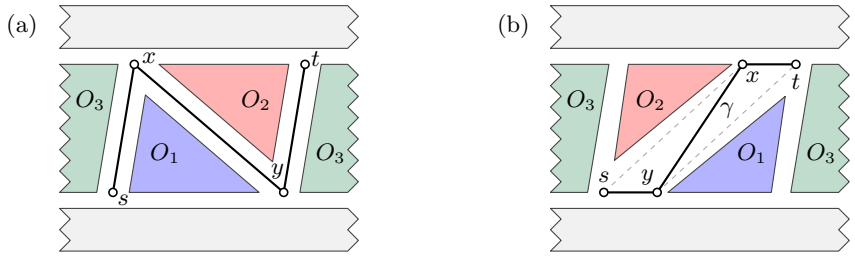


Figure 3 Eliminating the turns x and y in the proof of Lem. 4.2.

▶ **Theorem 3.2.** *It can be tested in $O(n \log^3 n)$ time whether a given n-vertex graph G admits a radial upward planar embedding with a given planar embedding $\mathcal{E}$.*

4 Level Planarity with Fixed Embedding

In this section, we investigate (radial) level planarity testing with fixed embedding. We start with version, where the (radial) level embedding is fixed; it is based on Lemma 2.2.

▶ **Lemma 4.1.** *It can be tested in linear time whether a (radial) level embedding of a level graph in proper from is (radial) level planar.*

Next, consider the slightly more relaxed version where we seek to decide whether a given radial upward planar embedding is radial level planar. The following "untangling" lemma is partially known [2, Lemma 3.4], [1, Claim 2], but only for non-radial level planar drawings.

▶ **Lemma 4.2.** *Let $G = (V, E)$ be a level graph, let Γ be a (radial) level planar drawing of G with (radial) upward embedding $\mathcal{E}$. Let $a, b \in V$ with $\ell(a) < \ell(b)$ be incident to a face f such that embedding (a, b) inside f yields a (radial) upward embedding $\mathcal{E}'$ of $G + (a, b)$. If there exists a curve γ inside the face f of Γ that connects a and b and does not intersect the levels $\ell(a)$ and $\ell(b)$, there exists a (radial) level planar drawing Γ' of $G + (a, b)$ with (radial) upward embedding $\mathcal{E}'$.*

Proof Sketch. Subdivide γ so that it crosses all levels at vertices. A vertex of γ is a *turn* if its two neighbors lie on the same level. If γ has no turns, it is an upward drawing of (a, b). Otherwise, there exist consecutive turns x, y, $\ell(x) > \ell(y)$ that minimize $\ell(x) - \ell(y)$. Modify the drawing to eliminate these two turns. Let s be the last vertex before x with $\ell(s) = \ell(y)$ and let t be the first vertex after y with $\ell(t) = \ell(x)$ on γ; see Fig. 3 (a). Between levels $\ell(s)$ and $\ell(t)$, γ separates Γ into three regions O_1, O_2, O_3. Exchange O_1 and O_2 without changing the upward embedding, and remove the turns x and y from γ; see Fig. 3 (b). Use Lem. 2.2 to show that this preserves radial upward planarity. The proof follows by induction. ◀

We can use Lem. 4.2 to insert edges into a graph while maintaining (radial) upward planarity such that all faces end up having size at most 4.

▶ **Lemma 4.3.** *Let $G = (V, E)$ be a level graph together with a (radial) upward planar embedding $\mathcal{E}$. We can compute in linear time a level graph G' with (radial) upward planar embedding $\mathcal{E}'$ whose faces have size at most 4 so that G has a (radial) level planar drawing with embedding $\mathcal{E}$ if and only if G' has a (radial) level planar drawing with embedding $\mathcal{E}'$.*

Proof Sketch. Assume w.l.o.g. that G is in star form. Insert edges into G and $\mathcal{E}$ until all faces have size at most 4 while maintaining (radial) level planarity. Let f be a face of $\mathcal{E}$ with

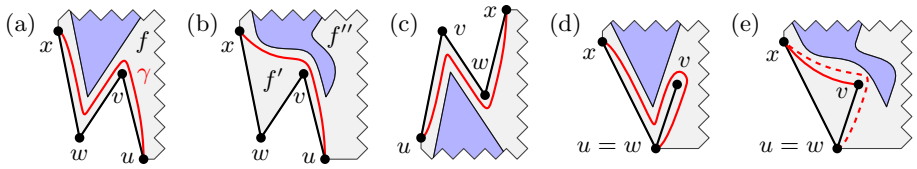


Figure 4 Inserting the edge (u, x) by defining the red curve γ (a) and using Lem. 4.2 to move the blue obstruction (b). The upward embedding and obstruction may look different (c). If w is a cutvertex, insert the edge (v, x) instead (e, d).

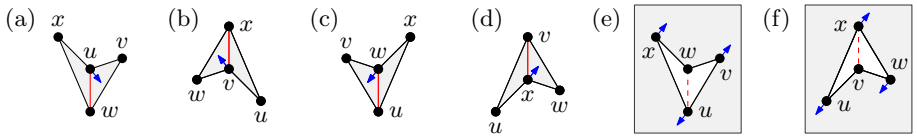


Figure 5 Inserting the red augmentation edges into inner faces (a–d) and the outer face (e, f).

at least five vertices incident to f. If there exist successive edges $(u, v), (v, w)$ or $(v, w), (u, v)$ on the facial walk of f, insert the edge (u, w) by closely following the successive edges within f. Otherwise each vertex incident to f is a face source or a face sink of f. Let (w, v) be an edge incident to f that minimizes $\ell(v) - \ell(w)$. Let (u, v) and (w, x) denote the edges that precede and succeed (w, v). Because (w, v) minimizes $\ell(v) - \ell(w)$ and because G is in star form, it is $\ell(u) \leq \ell(w) < \ell(v) \leq \ell(x)$. See Fig. 4. If $u \neq w$ and $v \neq x$, then $\ell(u) < \ell(w) < \ell(v) < \ell(x)$ and there is a curve γ from u to x such that by Lem. 4.2 (u, x) can be inserted into G and $\mathcal{E}$ while maintaining level planarity; see Fig. 4 (a–c). If $u = w$ or $v = x$, insert (v, x) or (u, w) instead; see Fig. 4 (d, e). ◀

▶ **Theorem 4.4.** *Let $G = (V, E)$ be a level graph together with a (radial) upward planar embedding $\mathcal{E}$. It can be tested in linear time whether G admits a (radial) level planar drawing with (radial) upward planar embedding $\mathcal{E}$.*

Proof Sketch. Lem. 4.3 lets us assume that each face has size at most 4. No source/sink is assigned to faces of size at most 3. Let f be an inner face of size 4 and let u, v, w, x denote the facial walk of f. Let u, w be face sources and let v, x be face sinks. See Fig. 5 (a–d). Exactly one of u, v, w, x is assigned to f. Bertolazzi et al. [5] show that if u is assigned to f, then the edge (w, u) can be inserted into f in any upward planar drawing. For level planarity, this requires $\ell(w) < \ell(u)$; see Fig. 5 (a). Thus, if $\ell(w) \geq \ell(u)$, then G is not (radial) level planar with (radial) upward embedding $\mathcal{E}$. The cases when v, w or x are assigned to f are symmetric; see Fig. 5 (b–d). The outer face can be handled similarly; see Fig. 5 (e, f). Inserting these edges gives a level graph G' with a single source u and a single sink x together with an embedding $\mathcal{E}'$. This implies that G is (radial) level planar with upward embedding $\mathcal{E}$. ◀

Finally, consider the least restricted version of level planarity testing, where only a planar embedding $\mathcal{E}$ of G is fixed. Our result relies on a translation of Lem. 4.3 to this setting.

▶ **Lemma 4.5.** *Let G be a level graph with a planar embedding $\mathcal{E}$. We can compute in linear time a level graph G' with planar embedding $\mathcal{E}'$ whose faces have size at most 4 so that G has a (radial) level planar drawing with embedding $\mathcal{E}$ if and only if G' has a (radial) level planar drawing with embedding $\mathcal{E}'$.*

▶ **Theorem 4.6.** *Let G be a level graph together with a planar embedding $\mathcal{E}$. It can be tested in linear time whether G admits a (radial) level planar drawing with planar embedding $\mathcal{E}$.*

Proof Sketch. Lem. 4.5 lets us assume that each face has size at most 4. Check whether there exists a (radial) upward planar embedding with planar embedding $\mathcal{E}$ using Lemmas 2.1 and 3.1. Exactly one source/sink is assigned to each inner face of size 4. From the assignment of a source or sink to an inner face f, we know which edge must be inserted in the augmentation procedure of Bertolazzi et al., and we can test beforehand, whether this respects the given level assignment. This leaves at most two vertices that can be assigned to f. Such an assignment can then be computed in linear time. ◀

References

1 Patrizio Angelini, Giordano Da Lozzo, Giuseppe Di Battista, and Fabrizio Frati. Strip planarity testing for embedded planar graphs. *Algorithmica*, 77(4):1022–1059, 2017. doi:10.1007/s00453-016-0128-9.

2 Esther M. Arkin, Michael A. Bender, Erik D. Demaine, Martin L. Demaine, Joseph S. B. Mitchell, Saurabh Sethia, and Steven Skiena. When can you fold a map? *Comput. Geom.*, 29(1):23–46, 2004. doi:10.1016/j.comgeo.2004.03.012.

3 Christopher Auer, Christian Bachmaier, Franz-Josef Brandenburg, and Andreas Gleißner. Classification of planar upward embedding. In Marc J. van Kreveld and Bettina Speckmann, editors, *Proceedings of the 19th International Symposium on Graph Drawing (GD'11)*, volume 7034 of *Lecture Notes in Computer Science*, pages 415–426. Springer, 2011. doi:10.1007/978-3-642-25878-7_39.

4 Christian Bachmaier, Franz-Josef Brandenburg, and Michael Forster. Radial level planarity testing and embedding in linear time. *J. Graph Algorithms Appl.*, 9(1):53–97, 2005. doi:10.7155/jgaa.00100.

5 Paola Bertolazzi, Giuseppe Di Battista, Giuseppe Liotta, and Carlo Mannino. Upward drawings of triconnected digraphs. *Algorithmica*, 12(6):476–497, 1994. doi:10.1007/BF01188716.

6 Glencora Borradaile, Philip N. Klein, Shay Mozes, Yahav Nussbaum, and Christian Wulff-Nilsen. Multiple-source multiple-sink maximum flow in directed planar graphs in near-linear time. *SIAM J. Comput.*, 46(4):1280–1303, 2017. doi:10.1137/15M1042929.

7 Guido Brückner and Vera Chekan. Drawing two posets. *CoRR*, abs/2010.12928, 2020. arXiv:2010.12928.

8 Guido Brückner, Ignaz Rutter, and Peter Stumpf. Level planarity: Transitivity vs. even crossings. In Therese C. Biedl and Andreas Kerren, editors, *Proceedings of the 26th International Symposium on Graph Drawing and Network Visualization (GD'18)*, volume 11282 of *Lecture Notes in Computer Science*, pages 39–52. Springer, 2018. doi:10.1007/978-3-030-04414-5_3.

9 Radoslav Fulek, Michael J. Pelsmajer, Marcus Schaefer, and Daniel Štefankovič. *Hanani-Tutte, Monotone Drawings, and Level-Planarity*, pages 263–287. Springer New York, New York, NY, 2013. doi:10/dxxr.

10 Harold N. Gabow. An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems. In David S. Johnson, Ronald Fagin, Michael L. Fredman, David Harel, Richard M. Karp, Nancy A. Lynch, Christos H. Papadimitriou, Ronald L. Rivest, Walter L. Ruzzo, and Joel I. Seiferas, editors, *Proceedings of the 15th Annual ACM Symposium on Theory of Computing*, pages 448–456. ACM, 1983. doi:10.1145/800061.808776.

11 Ashim Garg and Roberto Tamassia. On the computational complexity of upward and rectilinear planarity testing. *SIAM J. Comput.*, 31(2):601–625, 2001. doi:10.1137/S0097539794277123.

12 S. Mehdi Hashemi. Digraph embedding. *Discret. Math.*, 233(1-3):321–328, 2001. doi:10.1016/S0012-365X(00)00249-1.

13 S. Mehdi Hashemi, Ivan Rival, and Andrzej Kisielewicz. The complexity of upward drawings on spheres. *Order*, 14:327–363, 1997. doi:10.1023/A:1006095702164.

14 John E. Hopcroft and Robert Endre Tarjan. Efficient planarity testing. *J. ACM*, 21(4):549–568, 1974. doi:10.1145/321850.321852.

15 Michael Jünger, Sebastian Leipert, and Petra Mutzel. Level planarity testing in linear time. In Sue Whitesides, editor, *Graph Drawing, 6th International Symposium, GD'98, Montréal, Canada, August 1998, Proceedings*, volume 1547 of *Lecture Notes in Computer Science*, pages 224–237. Springer, 1998. doi:10.1007/3-540-37623-2_17.

Row-based Rectangle Contact Representations of Triangular Grid Graphs*

Martin Nöllenburg, Anaïs Villedieu, and Jules Wulms

Algorithms and Complexity Group, TU Wien, Vienna, Austria
{noellenburg, avilledieu, jwulms}@ac.tuwien.ac.at

Abstract

Semantic word or tag clouds are popular visualizations summarizing texts by displaying their most relevant keywords while also optimizing which keywords are placed next to each other so as to communicate some semantic information. When modeling words as their bounding boxes and semantic proximity as touching boxes, this type of problem translates into a rectangle contact graph optimization problem, where as many of the desired contacts should be realized while no undesired contacts occur. In this paper we study this contact optimization problem with rectangles of prescribed width and unit height (modeling words in a single font size), which are to be placed in multiple rows of a word cloud. We present algorithms and an ILP model to maximize the number of realized adjacencies or to minimize the total size of any gaps between words for the special case of a triangular grid graph representing the desired contacts.

1 Introduction

Contact representations of planar graphs are a well-studied topic in graph theory, graph drawing, and computational geometry [7,8,10]. Vertices are represented by geometric objects, e.g., disks or polygons, and two objects touch if and only if they are connected by an edge. They find many applications, for instance in VLSI design [18], cartograms [12], or semantic word clouds [1,17], the latter of which motivate this paper.

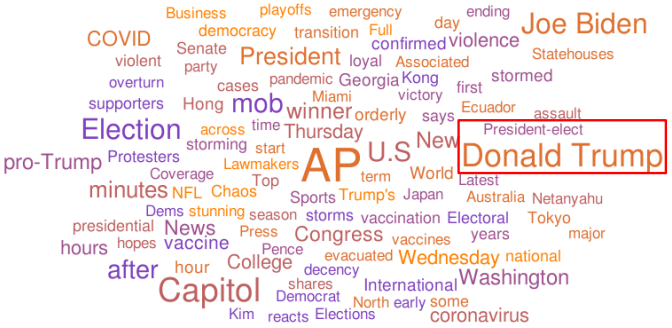


Figure 1 A word cloud generated from the apnews.com frontpage using worditout.com on the day of the certification of Joe Biden's victory.

Word or tag clouds are popular visualizations that summarize textual information in an aesthetically pleasing way. They show the main themes of a text by displaying the most important keywords obtained from text analysis and scale the word size to their frequency in the text. Word clouds became widespread after the first automated generation tool "Wordle"

* We acknowledge funding by the Austrian Science Fund (FWF) under grant P31119.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

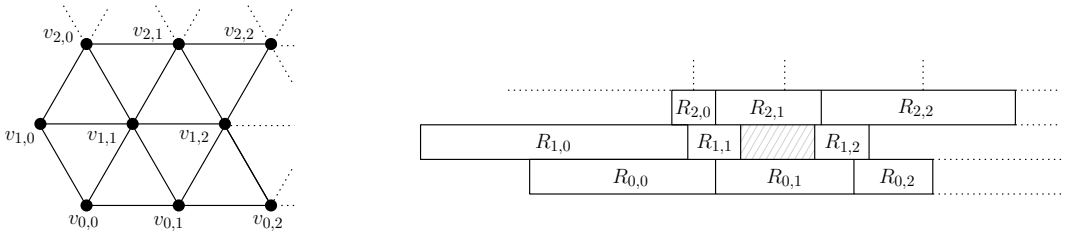


Figure 2 A 3×3 triangular grid and the associated representation

was published in 2009 [14]. Word clouds with their different font sizes and words packed without semantic context, such as the one shown in Figure 1, have also received some criticism as their audience sometimes fails at understanding the underlying data (while enjoying their playful nature) [9]. For example, neighboring words that are not semantically related can be misleading (see marked words in Figure 1). As a way to improve readability, *semantic* word clouds have been introduced [1, 6, 17]. In semantic word clouds, an underlying edge-weighted graph indicates the semantic relatedness of two words, whose positions are chosen such that semantically related words are next to each other while unrelated words are kept far apart.

Classic word clouds are often generated using forced-based approaches, alongside with a spiral placement heuristic [14–16] that allows for a very compact final layout. This method is powerful even when the general position of a word is dictated by an underlying map [4, 11]. Semantic word clouds on the other hand have been approached with many different techniques, e.g., force directed [6], seam-carving [17], and multi dimensional scaling [2]. The problem has also been studied from a theoretical point of view, where an edge of the semantic word graph is realized if the bounding boxes of two related words properly touch; the realized edge weight is gained as profit. Then the semantic word cloud problem can be phrased as the optimization problem to maximize the total profit. Barth et al. [1] and later Bekos et al. [3] gave several hardness and approximation results for this problem (and some variations) on certain graph classes. The underlying geometric problem also has links to more general contact graph representation problems, like rectangular layouts [5] or cartograms [13].

Grid graphs are missing from the current literature on rectangle contact graphs, yet they have some interesting properties in the context of word clouds. For instance, they are compact, have an even distribution of words and our eye naturally understands words grouped in lines or tables. In this paper we study grid-like and row-based contact graphs of unit-height but arbitrary-width rectangles, which may represent the bounding boxes of words with fixed font size, as a first step towards more complex settings. For example, our setting can easily be generalized to row-based layouts in which each vertex can have a different number of neighbors on the layers above and below itself. For grid-like and row-based layouts, we consider the *area minimization* and the *contact maximization* problems.

2 Preliminaries

Let $G = (V, E)$ be a triangular grid graph with L layers and K vertices per layer. Each vertex $v = v_{i,j} \in V$ is indexed by its position in the grid: $v_{i,j}$ with $i \in [0, L-1]$ and $j \in [0, K-1]$ is the j^{th} vertex on the i^{th} layer. We associate to each vertex an axis-aligned unit-height rectangle $R_{i,j}$ with width $w_{i,j}$, x-position $x_{i,j}$ given by the x-coordinate of its bottom left corner, and y-coordinate i (see Figure 2). Let $w_{\max} := \max\{w_{i,j} \mid v_{i,j} \in V\}$.

The rectangles are laid out in the plane, such that they do not overlap except on their boundaries. Such a layout $\mathcal{R}$ is called a *representation* of G. An edge $(u, v) \in E$ is *realized*

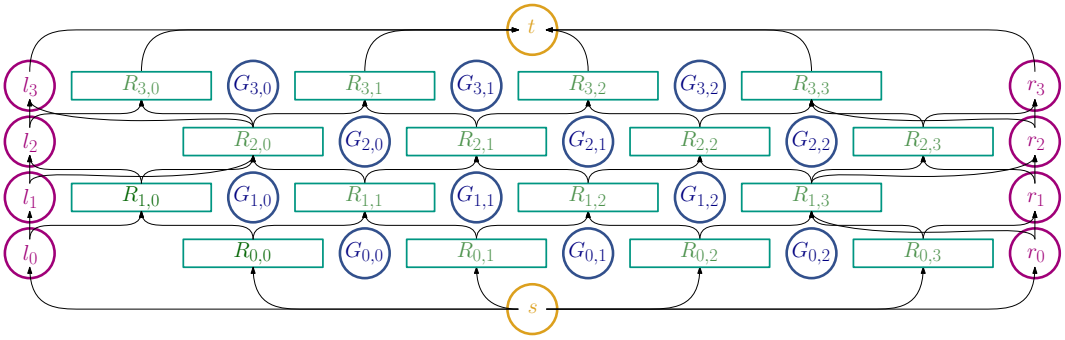


Figure 3 A sketch of the flow network on a 3×4 grid, edges incident to gaps are omitted.

in a representation $\mathcal{R}$ if rectangles R_u and R_v, representing vertices u and v, intersect along their boundaries for a positive length $\varepsilon > 0$, which we will denote by $(R_u, R_v) \in \mathcal{R}$. If R_u and R_v are on the same layer this happens along a vertical boundary and these contacts are called *horizontal contact*. Otherwise, the intersection is located along a horizontal boundary if R_u and R_v are on adjacent layers and these are *vertical contact*. Contacts between boxes whose vertices are not adjacent in G are *forbidden adjacencies*. If such adjacencies exist, then the representation $\mathcal{R}$ is invalid, as it could let a user infer that unrelated words are related. Within this model we study two problem variations, area minimization and contact maximization. Forbidden adjacencies are important considerations in both versions.

3 Area minimization

To a given vertex-weighted triangular grid graph $G = (V, E)$, we associate a flow network $N = (D = (V', E'); l; c; b; cost)$, with edge capacity lower bound $l\colon E' \to \mathbb{R}_0^+$, edge capacity $c\colon E' \to \mathbb{R}_0^+$, vertex production/consumption $b\colon V' \to \mathbb{R}$ and cost function $cost\colon E' \to \mathbb{R}_0^+$. Each unit of cost will represent a unit length gap and each unit of flow on an edge will represent a unit length contact. To create the network we create two vertices for each rectangle and one for each potential gap. Every edge that ends on a gap vertex has $cost = 1$. We also add an edge e between any two vertices of the same rectangle $R_{i,j}$ with $l(e) = c(e) = w_{i,j}$ and no cost to ensure that rectangle nodes receive exactly as much flow as they are wide. The global structure of the network is sketched in Figure 3; Figure 4 shows the complete local network around a pair of rectangle vertices.

The intuition behind the network is that it represents a stack of layers consisting of rectangles and gaps, with a width of $w_{\max} \cdot K$. Each rectangle is as wide as the amount of flow its representative vertex receives, and has contacts with its upper and lower neighbors as wide as the flow on the edges representing these contacts. Every vertex has edges to the layer above as far as their rectangle is allowed to reach: for example in an even layer, rectangle $R_{i,j}$ cannot have contact with $R_{i+1,j-1}$, so if $R_{i,j}$ is very wide, the flow might push $R_{i+1,j-1}$ to the left and widen the gap to the right of $R_{i+1,j-1}$. Every gap vertex has edges to the layer above as far as their left and right rectangle neighbors can reach: if these edges would reach farther, then the neighboring rectangles on the same layer could be pushed into forbidden adjacencies.

▶ **Theorem 3.1.** *Given a graph $G = (V, E)$, the minimum cost of flow $f = w_{\max} \cdot K$ in N equals the minimum total gap length of any grid contact representation of G. We can construct the corresponding area-minimal representation of G from f.*

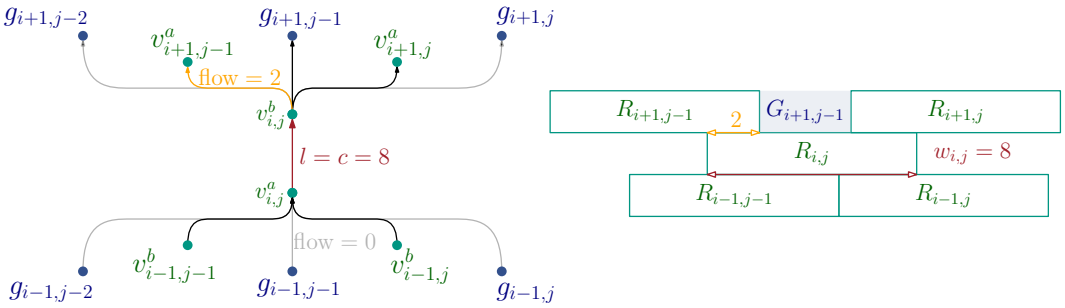


Figure 4 The network structure around rectangle $R_{i,j}$ on an odd layer.

From this network we can easily deduce that the minimum cost flow corresponds to the solution that minimizes total gap length: only gap vertices have a cost, hence the flow avoids these vertices whenever possible. In the solution to the network, the flow values found on each edge give us the length of the overlap between these elements, which allow us to construct the corresponding representation. On each graph layer with K nodes the network layer has $2 \cdot K + 1$ node, and each vertex has at most degree 6 (except $2K + 1$ for s), so we can construct the network and then solve minimum flow problem in polynomial time.

4 Contact Maximization

In this section we propose algorithms that maximize the number of realized contacts in the representation, while preventing forbidden adjacencies. We start with a linear-time algorithm for $L = 2$, followed by an integer linear programming (ILP) model for $L > 2$. The complexity for $L > 2$ remains open. Finally, we experimentally compare the flow network from Section 3 and the ILP on runtime, gapwidth and failed contacts.

4.1 Linear-time algorithm for $L = 2$

Before we describe our algorithm for a grid consisting of two layers, we first introduce a *block*: a maximal sequence of consecutive rectangles in a layer i, for which each horizontal contact is realized. A block from the jth until the lth vertex of row i is the sequence $(R_{i,j}, \ldots, R_{i,l})$, where for each $k \in [j, l-1]$ holds that $(R_{i,k}, R_{i,k+1}) \in \mathcal{R}$.

We propose the following greedy algorithm $\mathcal{A}$ to maximize the number of contacts for instances with two layers ($L = 2$). Algorithm $\mathcal{A}$ adds rectangles to representation $\mathcal{R}$ from left to right, starting with $R_{1,0}$ and alternating between the two layers: after a rectangle $R_{1,a}$ is added, it is followed by $R_{0,a}$, which in turn is followed by $R_{1,a+1}$. We call this ordering $\prec$.

Let rectangle $R_{i,j}$ be a new rectangle that is added to the representation, then we can make the following greedy choice. Initially, $R_{i,j}$ is placed as far left as possible, such that it realizes the (horizontal) contact with $R_{i,j-1}$. If $R_{i,j}$ does not extend at least ε past its predecessor in $\prec$ on the other layer, then slide the block containing $R_{i,j}$ rightwards until it does (see Fig. 5 top). If this sliding requires a previously established vertical contact to break, then we no longer insist on the horizontal contact $(R_{i,j-1}, R_{i,j})$. Instead, $R_{i,j}$ will start a new block, and it is placed such that it extends exactly ε past the last rectangle on the other layer (see Fig. 5 bottom). We then proceed to the next rectangle in $\prec$.

We show that $\mathcal{A}$ finds an optimal representation in linear time.

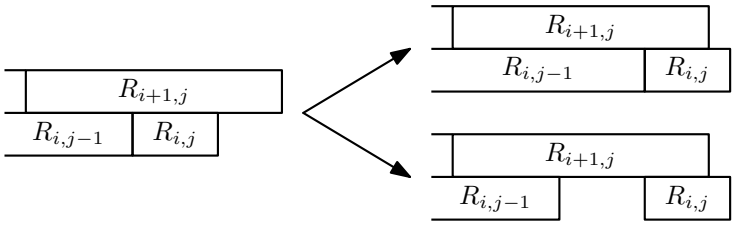


Figure 5 Greedy placement of $R_{i,j}$ with $i = 0$. At the top the block containing $R_{i,j}$ can slide to the right without breaking vertical contacts, while at the bottom $R_{i,j}$ starts a new block.

▶ **Lemma 4.1.** *There exists an optimal solution to the contact maximization problem, that realizes all vertical contacts.*

Proof sketch. We show that every optimal solution can be adapted to realize all vertical contacts, while not decreasing the number of realized contacts. Figure 6 gives an overview of interesting cases in the case distinction that shows how a solution should be adapted to ensure that rectangle $R_{0,j}$ realizes its vertical contacts. Note that the number of lost contacts (in red) never exceeds the number of gained contacts (in green/blue). ◀

To prove the next lemma, we use the *sliding range* of a block of rectangles. The sliding range is defined as the range in which the rectangles can move without changing the realized contacts. Since we have only two rows, the sliding range of a block is determined by how far a block can move left or right, before vertical contacts of the individual rectangles change.

▶ **Lemma 4.2.** *For representations $\mathcal{R}_1$ and $\mathcal{R}_2$, that have an optimal number of contacts and realize all vertical contacts, assume $\mathcal{R}_1$ contains a maximal block $B_1 = (R_{i,j}, \ldots, R_{i,k})$, while $\mathcal{R}_2$ contains maximal block $B_2 = (R_{i,l}, \ldots, R_{i,m})$. If $j \leq l$, then $k \leq m$.*

Proof. Assume that $k > m$, and consider both blocks B_1 and B_2 up to $R_{i,m}$, which we call B'_1 and B'_2 respectively. Since $j \leq l$, we know that the sliding range of B'_2 is larger than the sliding range of B'_1, as the rectangles in $B'_1 \setminus B'_2$ can only put more constraints on the sliding range. Therefore, B_2 can include all rectangles from $R_{i,m+1}$ to $R_{i,k}$ without breaking vertical contacts, contradicting the maximality of B_2. ◀

We can now argue that the greedy choice of $\mathcal{A}$ is optimal and $\mathcal{A}$ runs in linear time.

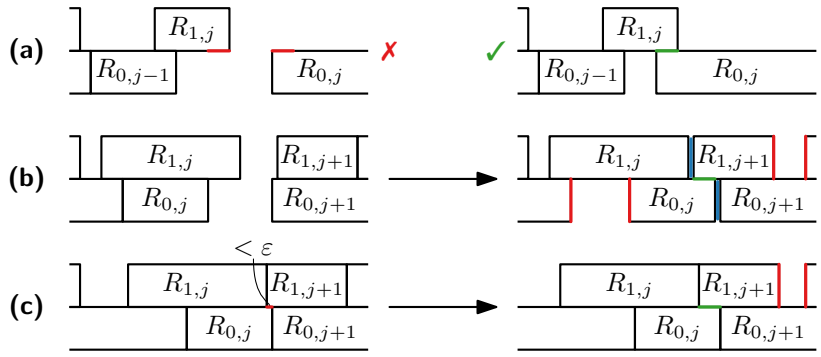


Figure 6 Three configurations where $R_{0,j}$ can realize vertical contacts with $R_{1,j}$ and $R_{1,j+1}$ without reducing the total number of contacts. New and lost contacts are shown in green and red respectively. In blue we show contacts that can be realized, if $R_{0,j}$ and $R_{0,j+1}$ extend equally far.

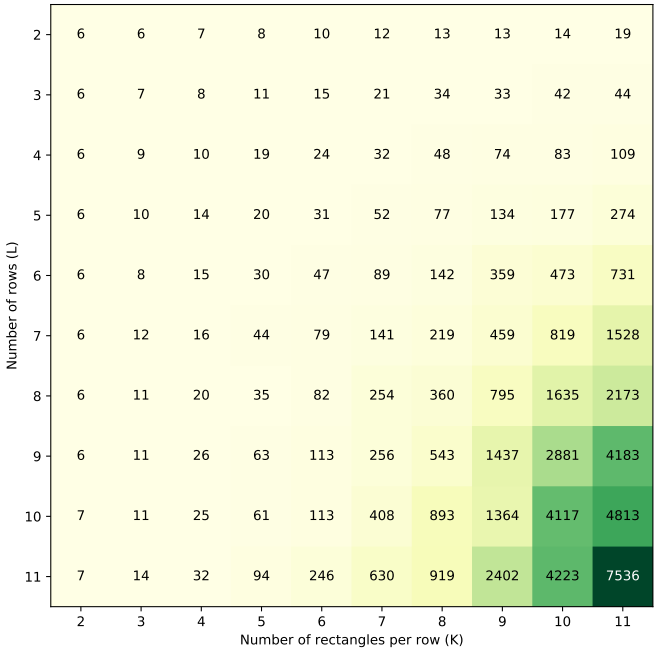


Figure 7 Average runtimes in ms for 30 runs of the ILP. Rectangle widths are randomly generated.

▶ **Theorem 4.3.** *Algorithm $\mathcal{A}$ maximizes the number of realized contacts in $O(n)$ time.*

Proof sketch. By Lemma 4.1, we know that there always exists an optimal representation that realizes all vertical contacts. Our greedy choice ensures that all vertical contacts are realized, and hence would lead to an optimal solution, if the number of blocks is minimal. We prove by induction that $\mathcal{A}$ finds the minimum number of blocks, and the smallest number of rectangles in the rightmost block of each row. The base case of a single block is trivial.

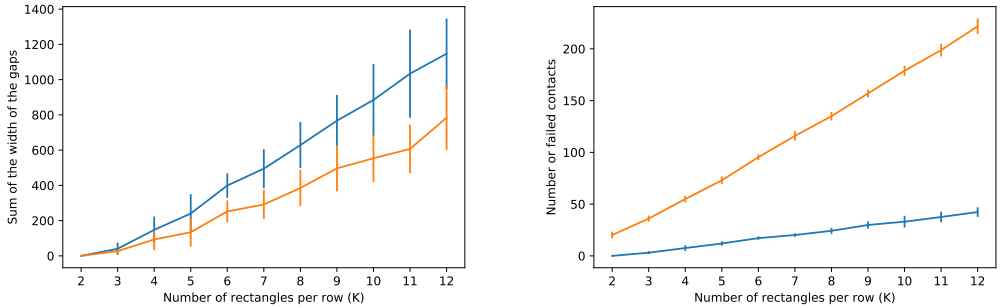
In the step case we apply Lemma 4.2, concluding that the rightmost blocks constructed by $\mathcal{A}$ can be extended with at least as many rectangles as in any optimal representation that realizes all vertical contacts. Furthermore, $\mathcal{A}$ always tries to extend the rightmost block and creates a new block only when this is unavoidable. Thus the number of blocks and number of rectangles in the rightmost blocks is minimal, and $\mathcal{A}$ computes an optimal solution.

Every rectangle R is considered only once when placed by $\mathcal{A}$. After placement, the constraints of R are stored in the sliding range of its block. Updating or initializing a sliding range takes constant time, hence we spend $O(1)$ time for each of the n rectangles. ◀

4.2 An Integer-Linear Program for $L > 2$

For the case of $L > 2$, we construct an ILP model. The intuition behind it is the following. We create a binary contact variable c for each desired rectangle contact. If a contact is lost by the position of the two involved rectangles, we must set $c = 1$ to satisfy all constraints; if the rectangles satisfy their contact constraints, we can set $c = 0$. Forbidden adjacencies are dealt with using hard constraints on the rectangle coordinates. The objective is to minimize the sum over all contact variables to maximumize the number of realized contacts in a solution.

The ILP computes solutions for instances with high K much slower than instances with high L, and solves 10×10 instances in a few seconds using Gurobi (see Figure 7). The network flow solves 10×10 instance on average in 168ms. Most word clouds should not



(a) Total width of gaps in solution

(b) Number of failed contacts in solution

Figure 8 Comparison of average gapwidth and failed contacts between the flow network (orange) and the ILP (blue), for 10-layer representations. Vertical lines show standard deviation over 20 runs.

have too many words so higher K and L values are not needed in practice. As expected we find that the flow network produces more compact drawings (see Figure 8a). Since the flow network only minimizes the overall gap area, it does not try to prevent horizontal contact loss. Similarly the flow network does not prioritize sending flow to both upward neighbors, over sending flow to just one neighbor, resulting in significant contact loss (see Figure 8b).

5 Conclusion

In this paper we studied rectangle contact layout optimization problems for restricted input graphs as a step towards more general row-based rectangle contact representations. Open problems include extensions to more irregular grid graphs as well as the complexity for contact maximization on three or more layers.

References

1 Lukas Barth, Sara Irina Fabrikant, Stephen G. Kobourov, Anna Lubiw, Martin Nöllenburg, Yoshio Okamoto, Sergey Pupyrev, Claudio Squarcella, Torsten Ueckerdt, and Alexander Wolff. Semantic word cloud representations: Hardness and approximation algorithms. In Alberto Pardo and Alfredo Viola, editors, *Proceedings of the 11th Latin American Theoretical Informatics Symposium (LATIN2014)*, volume 8392 of *LNCS*, pages 514–525. Springer, 2014. doi:10.1007/978-3-642-54423-1_45.

2 Lukas Barth, Stephen G. Kobourov, and Sergey Pupyrev. Experimental comparison of semantic word clouds. In Joachim Gudmundsson and Jyrki Katajainen, editors, *Proceedings of the 13th International Symposium on Experimental Algorithms (SEA 2014)*, LNCS, pages 247–258. Springer International Publishing, 2014. doi:10.1007/978-3-319-07959-2_21.

3 Michael A. Bekos, Thomas C. van Dijk, Martin Fink, Philipp Kindermann, Stephen Kobourov, Sergey Pupyrev, Joachim Spoerhase, and Alexander Wolff. Improved approximation algorithms for box contact representations. *Algorithmica*, 77(3):902–920, 2017. doi:10.1007/s00453-016-0121-3.

4 Kevin Buchin, Daan Creemers, Andrea Lazzarotto, Bettina Speckmann, and Jules Wulms. Geo word clouds. In *Proceedings of the 9th IEEE Pacific Visualization Symposium (PacificVis 2016)*, pages 144–151, 2016. doi:10.1109/PACIFICVIS.2016.7465262.

5 Adam L. Buchsbaum, Emden R. Gansner, Cecilia M. Procopiuc, and Suresh Venkatasubramanian. Rectangular layouts and contact graphs. *ACM Transactions on Algorithms*, 4(1):8:1–8:28, 2008. doi:10.1145/1328911.1328919.

6 Weiwei Cui, Yingcai Wu, Shixia Liu, Furu Wei, Michelle X. Zhou, and Huamin Qu. Context preserving dynamic word cloud visualization. In *Proceedings of the 3rd IEEE Pacific Visualization Symposium (PacificVis 2010)*, pages 121–128, 2010. doi:10.1109/PACIFICVIS.2010.5429600.

7 Hubert de Fraysseix, Patrice Ossona de Mendez, and Pierre Rosenstiehl. On triangle contact graphs. *Combinatorics, Probability and Computing*, 3:233–246, 1994. doi:10.1017/S0963548300001139.

8 Stefan Felsner. Rectangle and square representations of planar graphs. In János Pach, editor, *Thirty Essays on Geometric Graph Theory*, pages 213–248. Springer, 2013. doi:10.1007/978-1-4614-0110-0_12.

9 Marti A. Hearst, Emily Pedersen, Lekha Patil, Elsie Lee, Paul Laskowski, and Steven Franconeri. An evaluation of semantically grouped word cloud designs. *IEEE Transactions on Visualization and Computer Graphics*, 26(9):2748–2761, 2020. doi:10.1109/TVCG.2019.2904683.

10 Paul Koebe. Kontaktprobleme der konformen abbildung. *Ber. Sächs. Akad. Wiss. Leipzig, Math.-Phys. Klasse*, 88:141–164, 1936.

11 Chenlu Li, Xiaoju Dong, and Xiaoru Yuan. Metro-wordle: An interactive visualization for urban text distributions based on wordle. *Visual Informatics*, 2(1):50–59, 2018. doi:10.1016/j.visinf.2018.04.006.

12 Sabrina Nusrat and Stephen Kobourov. The state of the art in cartograms. *Computer Graphics Forum*, 35(3):619–642, 2016. doi:10.1111/cgf.12932.

13 Marc van Kreveld and Bettina Speckmann. On rectangular cartograms. *Computational Geometry*, 37(3):175–187, 2007. doi:10.1016/j.comgeo.2006.06.002.

14 Fernanda B. Viegas, Martin Wattenberg, and Jonathan Feinberg. Participatory visualization with wordle. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1137–1144, 2009. doi:10.1109/TVCG.2009.171.

15 Yunhai Wang, Xiaowei Chu, Chen Bao, Lifeng Zhu, Oliver Deussen, Baoquan Chen, and Michael Sedlmair. Edwordle: Consistency-preserving word cloud editing. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):647–656, 2018. doi:10.1109/TVCG.2017.2745859.

16 Yunhai Wang, Xiaowei Chu, Kaiyi Zhang, Chen Bao, Xiaotong Li, Jian Zhang, Chi-Wing Fu, Christophe Hurter, Oliver Deussen, and Bongshin Lee. Shapewordle: Tailoring wordles using shape-aware archimedean spirals. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):991–1000, 2020. doi:10.1109/TVCG.2019.2934783.

17 Yingcai Wu, Thomas Provan, Furu Wei, Shixia Liu, and Kwan-Liu Ma. Semantic-preserving word clouds by seam carving. *Computer Graphics Forum*, 30(3):741–750, 2011. doi:10.1111/j.1467-8659.2011.01923.x.

18 Kok-Hoo Yeap and Majid Sarrafzadeh. Floor-planning by graph dualization: 2-concave rectilinear modules. *SIAM J. Comput.*, 22(3):500–526, 1993. doi:10.1137/0222035.

Recognition of Unit Disk Graphs for Caterpillars, Embedded Trees, and Outerplanar Graphs*

Sujoy Bhore[1], Soeren Nickel[2], and Martin Nöllenburg[2]

1 **Université Libre de Bruxelles**
`sujoy.bhore@gmail.com`
2 **TU Wien**
`[soeren.nickel|noellenburg]@ac.tuwien.ac.at`

Abstract

Unit disk graphs are graphs that have a *unit disk intersection representation* (UDR). In the recognition problem the objective is to decide whether a given graph is a unit disk graph, which is known to be NP-hard, even for planar graphs. In this work, we show that the recognition of unit disk graphs remains NP-hard for outerplanar graphs and for embedded trees.

1 Introduction

The representation of graphs as contacts or intersections of unit disks has been a major topic of investigation in geometric graph theory. A set of unit disks in $\mathbb{R}^2$ is a unit disk intersection representation (UDR) of a graph $G = (V, E)$, if there is a bijection between V and the set of unit disks such that two disks intersect if and only if they are adjacent in G. *Unit disk graphs* are graphs that admit UDR. *Unit disk contact graphs* (also known as penny graphs) are the subfamily of unit disk graphs that have a UDR with interior-disjoint disks, also called a *unit disk contact representation* (UDC). The famous circle packing theorem states that every planar graph has a contact representation by touching disks and vice versa [15]. Since then, a large body of research has been devoted to the representation of planar graphs as a contacts or intersections of geometric objects [5, 6, 10, 11].

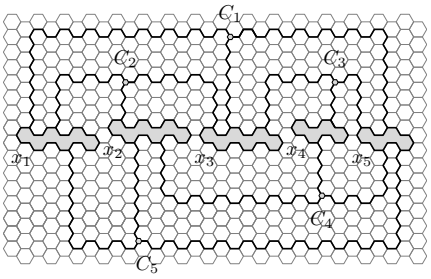
The recognition problem, where the objective is to decide whether a given graph admits a UDR, has a rich history [2, 3, 12, 13]. Breu and Kirpatrick [4] proved that it is NP-hard to decide whether a graph G admits a UDR or a UDC. Klemz et al. [14] showed that recognizing outerplanar unit disk contact graphs is already NP-hard, but decidable in linear time for caterpillars, i.e., trees whose internal nodes form a path. Eades and Wormald [9] showed that it is NP-hard to decide whether a given tree is a subgraph of a unit disk contact graph.

Recognition with a fixed embedding is an important variant of the recognition problem. Given a plane graph G, the objective in this problem is to decide whether G is a contact graph of interior-disjoint unit disks in the plane with the same cyclic order of neighbors at each vertex. Some recent works investigated the recognition problem of UDC with/without fixed embedding, and narrowed down the precise boundary between hardness and tractability; see [2, 7, 8].

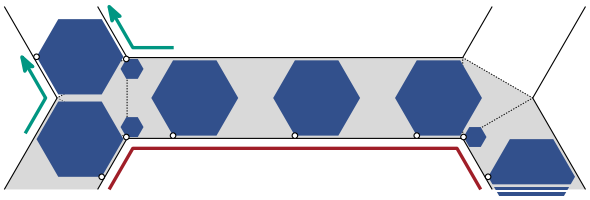
Most of the existing hardness results either rely on graphs with a large number of cycles, as also noted by Bowen et al. [2], or uses the fixed embedding to enforce certain representations [2, 7]. Therefore, an important open question in this area is to show the complexity of the recognition problem for non-embedded trees.

* The authors want to thank Maarten Löffler, Jonas Cleve, and Man-Kwun Chiu for fruitful discussions about the project during their research visits in Vienna.

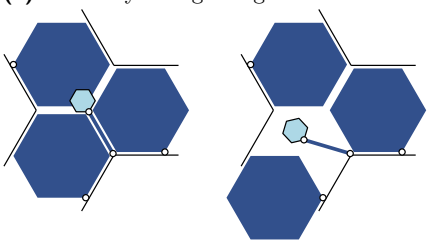
37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.



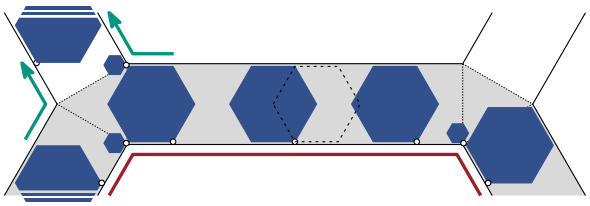
(a) Auxiliary hexagonal grid structure



(b) True state of variable cycle



(c) Unsatisfied (left) and one possible satisfied state (right) of clause gadget



(d) False state of variable cycle. Dotted hexagon indicates overlap

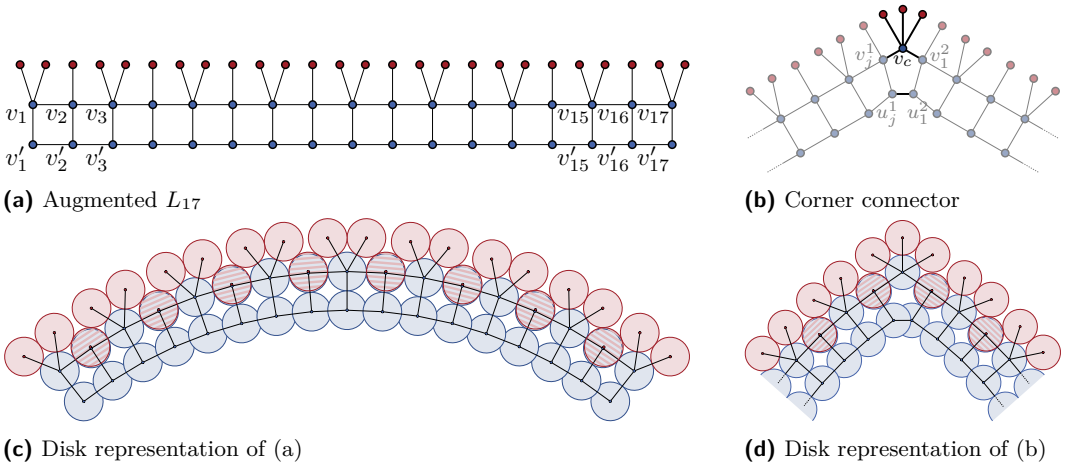
Figure 1 Auxiliary structure used by Bowen et al. [2], recreated. The incidence graph is embedded on a hexagonal grid (a). The edges are short corridors in which the blue hexagons are fitted, hinged at white vertices. Hexagons in the variable cycle (red line, grey backdrop) have two states (b) and (d). The clause gadget (c) requires one hexagon, which does not enter the junction.

Our Contribution. We show that recognizing unit disk graphs is NP-hard for outerplanar graphs and for embedded trees (Section 2), but decidable in linear time for caterpillars (Section 3). Note that due to space constraints, these results are only sketched here. We refer to the extended version [1] for a complete description of the results.

2 Hardness Results

Bowen et al. [2] proved that recognizing unit disk contact graphs is NP-hard for embedded trees, via a reduction from planar 3-SAT, which uses an auxiliary construction formulated as a realization of a polygonal linkage. A polygonal linkage is a set of polygons, which are realizable if they can be placed in the plane, s.t., predefined sets of points on the boundary of these points are identified. Bowen et al. define a set of hexagons in a hexagonal tiling (Fig. 1a) with small gaps between them, which form a hexagonal grid, in which a representation of the incidence graph of the planar 3-SAT instance is fitted. Smaller hexagons are fitted into cycles in this grid, s.t., they admit only two different realizations, see Fig. 1b, and determine the state of neighboring small hexagons, see Fig. 1d. The cycles represent variables in a true or false state. The states of the cycles are transmitted via chains of smaller hexagons in the gaps. The vertex, where three such chains meet, contains a small hexagon on a thin connection, which can only be realized if at least one transmitted state is *true*, see Fig. 1c. The polygonal linkage is realizable only if the planar 3-SAT instance was satisfiable. For a detailed description, we refer to Bowen et al. [2].

The building blocks of this reduction are hexagons of variable sizes and short segments, which are approximated with UDCs by Bowen et al. [2] by creating graphs, whose UDC must be within a constant Hausdorff distance of hexagons and long thin rhombi. We extend their notion of λ-stable approximations to UDRs.



(a) Augmented L_{17} **(b)** Corner connector

(c) Disk representation of (a) **(d)** Disk representation of (b)

Figure 2 Graph (a), (b) and disk (c), (d) representations of the ladder and corner connector, used to create the outline of the λ-stable approximations G_k^H and G_k^R. The bends in the UDRs are required but exaggerated.

▶ **Definition 2.1.** A graph G is a λ-stable approximation of a polygonal shape P if, in every UDR of G, a congruent copy of P and the union of all unit disks in the UDR have at most a Hausdorff distance of λ.

2.1 Outerplanar Graphs

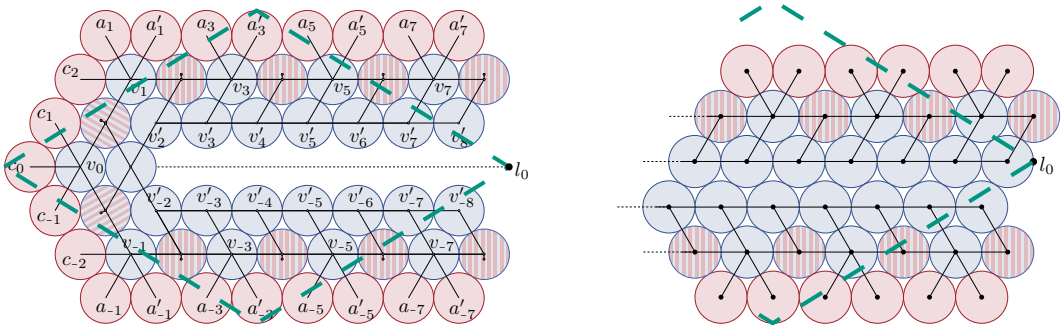
▶ **Theorem 2.2.** *Recognizing unit disk graphs is* NP*-hard for outerplanar graphs.*

We will prove Theorem 2.2 by providing outerplanar graphs G_k^H and G_k^R, which are $O(1)$-stable approximations of a hexagon and a rhombus, respectively. Then the NP-hardness follows immediately from the construction of Bowen et al. [2] sketched above. To obtain the $O(1)$-stable approximations we will first present two graphs, which enforce local bends in one direction of at least π and $\frac{4\pi}{3}$ in their UDR.

First a ladder L_k (see Figs. 2a and 2c) is a chain of pairwise connected vertices v_i and v'_i, also called the *outer* and *inner* vertices, respectively. Additionally so-called *extension neighbors*, which are connected to only one outer vertex each, are added on one side of the ladder, s.t., the outer vertices have alternating degrees of four and five. Since the ladder consists of a chain of C_4's, these neighbors are forced to be placed all on the outside. The minimal height of such a ladder is $2\sqrt{3}+2-\varepsilon$, which is the height of the smallest bounding box of a tight packing of three rows of unit circles minus a small constant ε.

The permission of overlap between adjacent neighbors allows for a placement of one neighbor almost on top of its adjacent outside vertex, which leads to an ever so slight inwards bend and, more importantly, any outwards bend is impossible. In order to force an inward bend of at least $\frac{4\pi}{3}$, we connect last inner vertex u_j^1 of one ladder with the first inner vertex u_1^2 of a second ladder and the last and first outer vertex v_j^1/v_1^2 of the first and second ladder respectively both with a vertex v_c, which has three attached extension neighbors, see Fig. 2b. This construction is called a corner connector. Since it is impossible to place the disk of any extension neighbor of v_c inside the 5-cycle $d(v_j^1), d(u_j^1), d(u_1^2), d(v_1^2), d(v_c)$, without overlapping at least two disks in the UDR, all extension neighbors are still forced to the outside and therefore $\measuredangle v_j^1 v_c v_1^2 > \frac{4\pi}{3}$, see Fig. 2d.

By placing two ladders opposite of each other and connecting them on one end with



(a) Widest possible position **(b)** Narrowest possible position

Figure 3 A 7-stable approximation G_7^R of a rhombus (dashed green line) superimposed on its UDR in its widest (a) and narrowest (b) configuration. Both UDRs require a small inward bend to be valid and hatched disks indicate almost overlapping placement of a red disk on a blue disk, with a small shift to the outside. Both inward bends and outward shifts are omitted.

three corner connectors as shown in Fig. 3, we create a 12-stable approximation G_k^R of a thin rhombus.

▶ **Lemma 2.3.** *For every integer k the outerplanar graph G_k^R in Fig. 3 is a 7-stable approximation of a rhombus of width $2k+6$ and height $6\sqrt{3}+2$.*

Proof Sketch. The graph G_k^R is made up of components, which make any bend to the outside impossible, see Fig. 3a. Since two ladders need to be overlap free, the minimum height of the ladders guarantees that part of the boundary of the union over all disks in the UDR of G_k^R lies above and below the line $\overline{c_0 l_0}$, even in its narrowest position, see Fig. 3b. The biggest vertical distance of the rhombus to this line is $3\sqrt{3}+1 \approx 6.196 < 7$. ◀

The construction of a 5-stable approximation G_k^H of a hexagon of side length $2k-1$ uses again ladders and corner connectors to trace the outline of the hexagon. Then the inside of this construction is filled with a set of ladders until no additional ladders can be added, see Fig. 4a. Outwards bends are impossible by construction of the ladders and corner connectors and inwards bends are strongly limited since the interior of the hexagon is almost completely filled with ladders. Since the amount of compression in a ladder is very limited, this leaves only a constant amount of space on the inside of a UDR of G_k^H. The result follows.

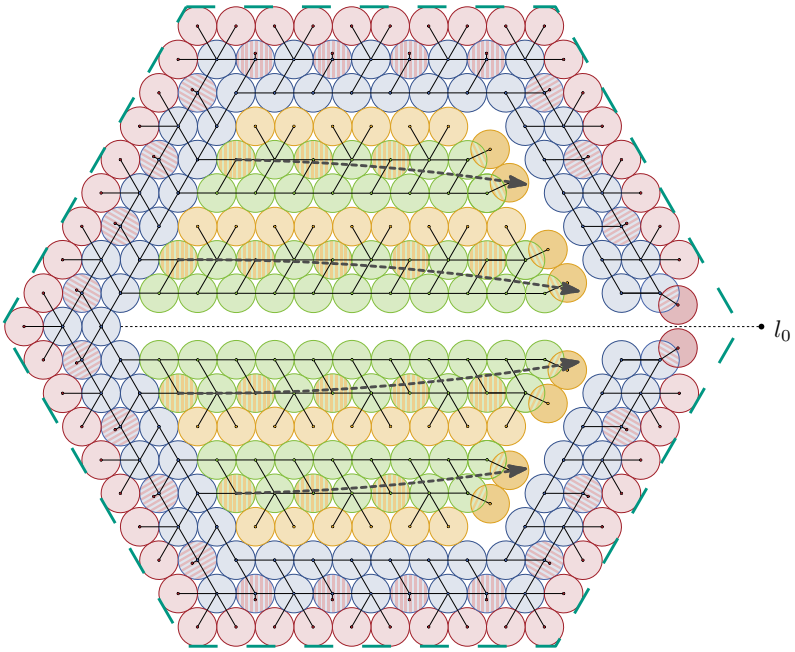
2.2 Trees with Embeddings

By slightly adapting the construction of the previous section, we can prove that recognizing unit disk graphs is NP-hard for embedded trees. The corresponding constructions are shown in Figs. 5 and 6.

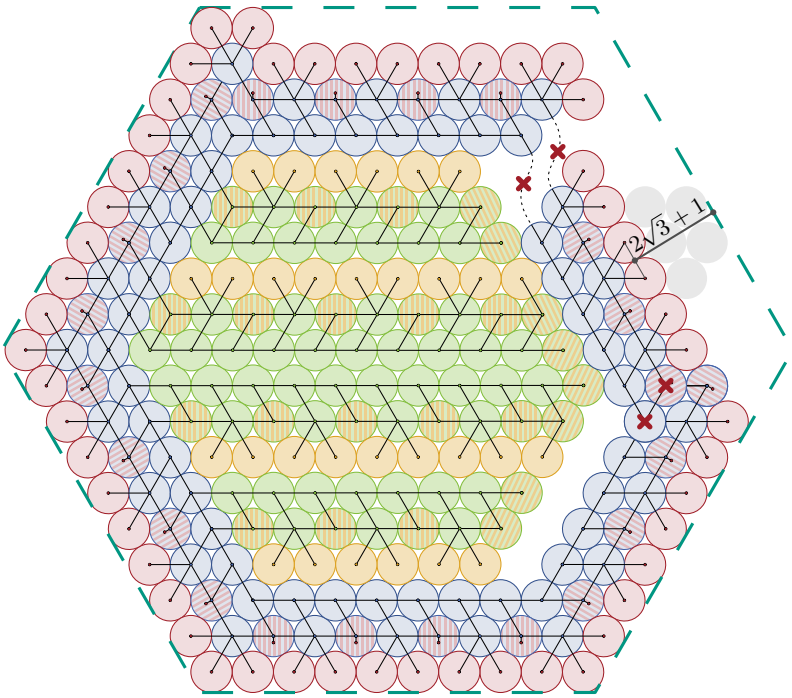
▶ **Theorem 2.4.** *Recognizing unit disk graphs is NP-hard for embedded trees.*

3 Recognition Algorithm for Caterpillars

We propose a linear-time algorithm using similar ideas to Klemz et al. [14], that recognizes if an input caterpillar graph $G = (V, E)$ admits a UDR or not, and provides the representation if one exists. However, we need to address several issues as a larger class of graphs admits a UDR compared to a UDC. Note, the input graph $G = (V, E)$ is essentially a tree whose

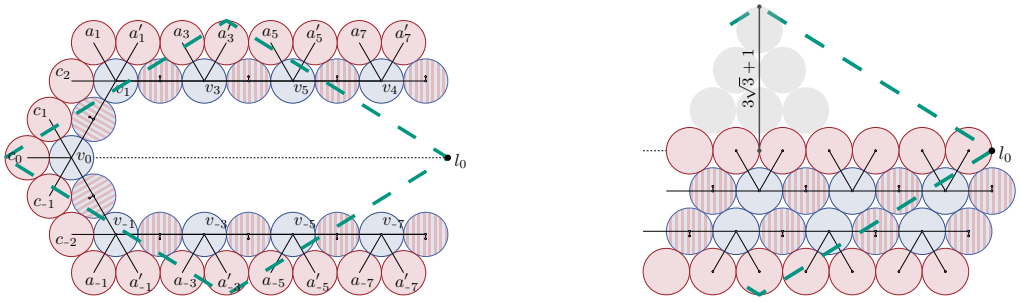


(a) A 5-stable approximation G_k^H of a hexagon (dashed green line) superimposed on its UDR. Necessary infinitesimal bends are omitted. The direction of these bends of the inner components is indicated with the grey arrow.



(b) In the narrowest position, the UDR of a G_k^H can – similar to the rhombus – fold all inner arms to one side (here to the lower side), and theoretically, there is space to fold the outer arms inward as depicted. The UDR of the arms are however again pairwise overlap free and the largest reachable distance any point on the boundary of the union of disks can have from the approximated regular hexagon is $2\sqrt{3}+1$. Note further that the folding of the rightmost arm is not possible as depicted, indicated by the red crossings, which mark areas of forced overlap and connections in the graph, which cannot be realized in a UDR. However in a setting with an extremely long arm, the small inward bend could lead to almost parallel arms, which run very close to each other, hence we analyze the (unreachable) worst case.

Figure 4 Construction and analysis of G_k^H



(a) T_k^R in widest position. **(b)** T_k^R in narrowest position.

Figure 5 A 7-stable approximation T_k^R of a long thin rhombus superimposed on its UDR in its widest (a) and narrowest (b) possible position. In both cases at any point along c_0l_0 at least one point on the boundary of the union of all disks in a UDR of T_k^R lies on or above c_0l_0 and on or below c_0l_0.

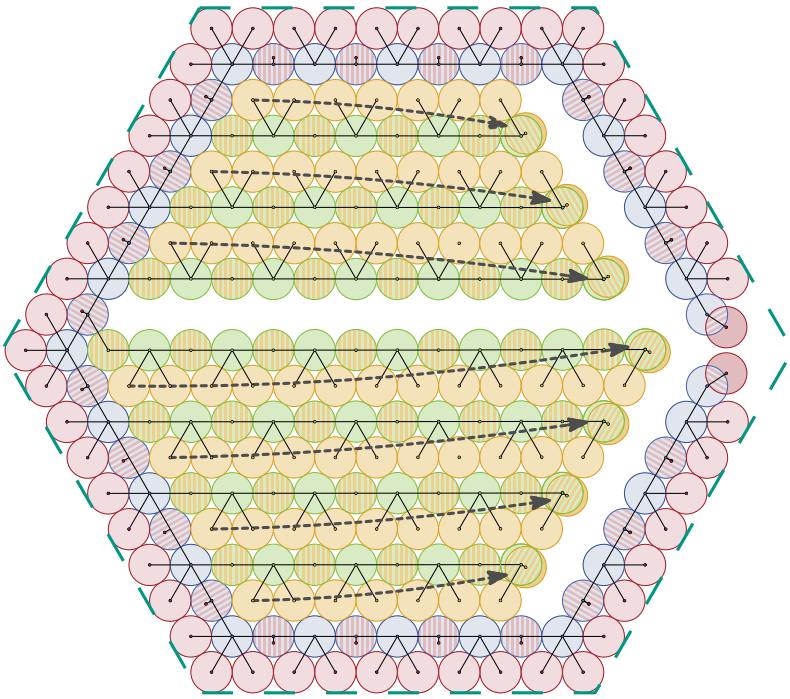
internal nodes form a backbone path $B_P = \{v_1, \ldots, v_k\}$ for some $k \in \mathbb{N}$. Observe that, if G contains a vertex of degree 6, then due to the disk packing property, it does not admit a UDR. Therefore, we know that every realizable caterpillar must have the maximum degree $\Delta \leq 5$. Moreover, it is easy to observe that all caterpillars with $\Delta \leq 4$ admit a UDC (and thus a UDR), as also noted by Klemz et al. [14]. However, not every caterpillar with $\Delta = 5$ is realizable as UDR. We show that two consecutive degree-5 vertices on B_P cannot be realized. The following lemma characterizes a "No" instance for the algorithm presented in Section 3.

▶ **Lemma 3.1.** *If B_P contains two adjacent degree 5 vertices u, v, then it does not admit a unit disk intersection representation.*

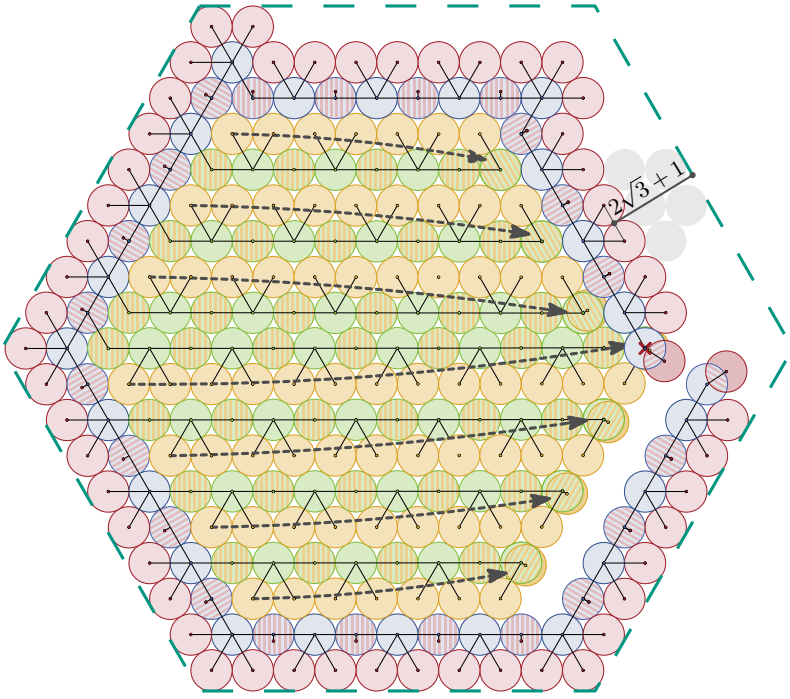
3.1 The Algorithm.

As a preprocessing step we augment all spine vertices of degree 3 or lower with additional degree 1 neighbors, s.t., they have degree 4. Now, consider a chain of vertices $v_i, \ldots, v_k$, all of degree 4. We place their disks on a horizontal line. The two neighboring leaf disks of a disk $d(v_j), i \leq j \leq k$ are placed one at the top and one at the bottom, respectively, s.t., the connection of their centers and the center $d(v_j)$ have an angle of $\frac{\pi}{3}$ plus a very small but continuously increasing value with the x-axis, see Fig. 7a. This way the angle between the center of the two neighbors and the center of $d(v_j)$ stays just slightly larger than $\frac{2\pi}{3}$ at all times. When we encounter a vertex $u = v_{k+1}$ of degree 5, we place the center of $d(u)$ also on the horizontal line and two leaf neighbors are placed on opposite sides. The third leaf is placed w.l.o.g. at the top, with an angle of $\frac{\pi}{3}$ plus a very small value to the previously placed leaf, see Fig. 7b.

If the following vertex $x = v_{k+2}$ has also degree 5, then the graph is not representable (see Lemma 3.1). Otherwise, we place the center of $d(x)$ again with an offset of $\frac{\pi}{3}$ plus a very small value to the last placed leaf neighbor (i.e, the third leaf of u). The direction of $\overline{ux}$ is now considered to be the new extension direction. The lower leaf is placed as planned, however the upper leaf of x is placed almost on top of $d(x)$, with a very small offset orthogonal to $\overline{ux}$, see Fig. 7b. Finally, the next disk $d(v_{k+3})$ is placed in the new extension direction touching $d(x)$ and two of its leaf neighbors can be placed with an angle just over $\frac{\pi}{3}$ relative to the new extension direction, i.e., an angle of just over $\frac{2\pi}{3}$ between them, see Figs. 7c and 7d. Note that at this point, if v_{k+3} has degree 5, we can immediately repeat this procedure.

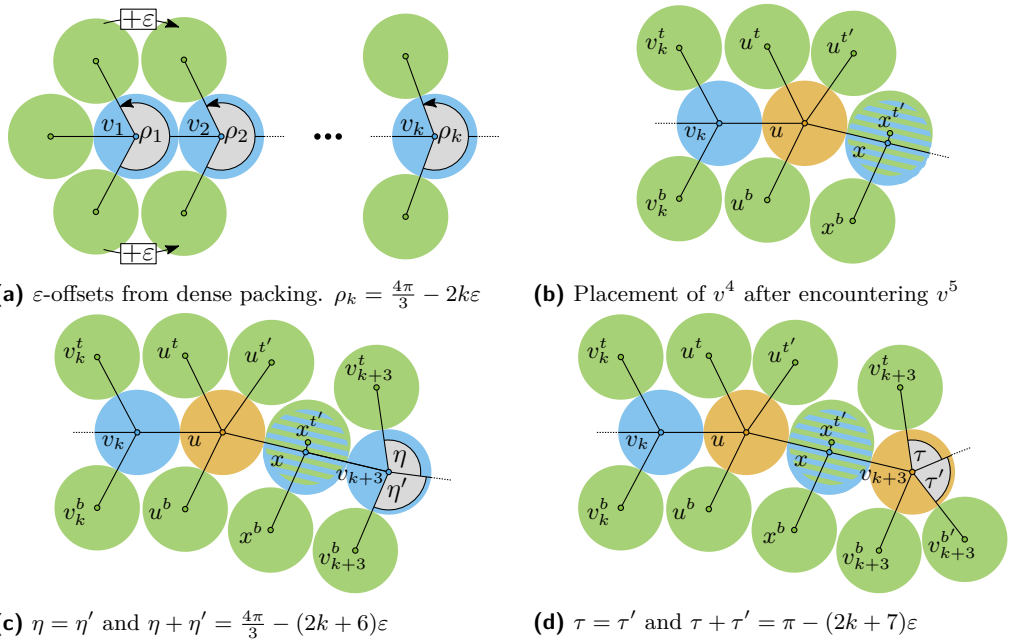


(a) A 5-stable approximation T_k^H of a hexagon superimposed on its UDR. Note that T_k^H is a tree. Visualization is analogue to Fig. 4a.



(b) The narrowest position of a T_k^H. The arguments for the distance to the approximated hexagon are very similar to Fig. 4b.

Figure 6 Construction and analysis of G_k^H



(a) ε-offsets from dense packing. $\rho_k = \frac{4\pi}{3} - 2k\varepsilon$ **(b)** Placement of v^4 after encountering v^5

(c) $\eta = \eta'$ and $\eta + \eta' = \frac{4\pi}{3} - (2k+6)\varepsilon$ **(d)** $\tau = \tau'$ and $\tau + \tau' = \pi - (2k+7)\varepsilon$

Figure 7 Chains of degree 4 are placed in a dense packing formation with small offsets (a). A degree five vertex places an additional leaf on one side (b). The following vertex of degree at most 4 places its leaf on the same side almost on top of itself. The next disk $d(x)$ can again be placed with the desired angle of just over $\frac{2\pi}{3}$ between two neighbors. Placement of $d(x)$ is possible if its degree 4 (c) or 5 (d).

As a final postprocessing step, we remove all degree 1 vertices that were added in the preprocessing step. Note that by placing the third neighbor of the first spine vertex of degree 5 at the top, and then alternatingly placing the next two at the bottom, then again two at the top and so on, the new extension direction returns to being horizontal after every second spine vertex of degree 5. In between it has alternatingly only a slight positive or negative angle offset relative to the x-axis. From the above description of the algorithm and the correctness analysis (refer to section 4 in [1]) we conclude the following theorem.

▶ **Theorem 3.2.** *Given a caterpillar graph $G = (V, E)$, it can be decided in linear time whether G is a unit disk graph. Moreover, if an unit disk intersection representation exists then it can be constructed in linear time.*

From Lemma 3.1 and Theorem 3.2, we get the following corollary.

▶ **Corollary 3.3.** *Let $G = (V, E)$ be a caterpillar graph. G admits a unit disk intersection representation if and only if G does not contain any two adjacent degree 5 vertices.*

References

1 Sujoy Bhore, Soeren Nickel, and Martin Nöllenburg. Recognition of unit disk graphs for caterpillars, embedded trees, and outerplanar graphs, 2021. arXiv:2103.08416.

2 Clinton Bowen, Stephane Durocher, Maarten Löffler, Anika Rounds, André Schulz, and Csaba D. Tóth. Realization of simply connected polygonal linkages and recognition of unit disk contact trees. In *Graph Drawing and Network Visualization (GD'15)*, volume 9411 of *LNCS*, pages 447–459. Springer International Publishing, 2015. doi:10.1007/978-3-319-27261-0_37.

3 Heinz Breu and David G. Kirkpatrick. On the complexity of recognizing intersection and touching graphs of disks. In Franz J. Brandenburg, editor, *Graph Drawing (GD'95)*, volume 1027 of *LNCS*, pages 88–98. Springer, 1996. doi:10.1007/BFb0021793.

4 Heinz Breu and David G. Kirkpatrick. Unit disk graph recognition is NP-hard. *Comput. Geom. Theory Appl.*, 9(1–2):3–24, 1998. doi:10.1016/S0925-7721(97)00014-X.

5 Jérémie Chalopin, Daniel Gonçalves, and Pascal Ochem. Planar graphs have 1-string representations. *Discrete & Computational Geometry*, 43(3):626–647, 2010. doi:10.1007/s00454-009-9196-9.

6 Steven Chaplick and Torsten Ueckerdt. Planar graphs as VPG-graphs. In Walter Didimo and Maurizio Patrignani, editors, *Graph Drawing (GD'12)*, volume 7704 of *LNCS*, pages 174–186. Springer, 2013. doi:10.1007/978-3-642-36763-2_16.

7 Man-Kwun Chiu, Jonas Cleve, and Martin Nöllenburg. Recognizing embedded caterpillars with weak unit disk contact representations is NP-hard, 2020. arXiv:2010.01881.

8 Jonas Cleve. Weak unit disk contact representations for graphs without embedding, 2020. arXiv:2010.01886.

9 Peter Eades and Nicholas C. Wormald. Fixed edge-length graph drawing is np-hard. *Discrete Applied Mathematics*, 28(2):111–134, 1990. doi:10.1016/0166-218X(90)90110-X.

10 Stefan Felsner. Rectangle and square representations of planar graphs. In János Pach, editor, *Thirty Essays on Geometric Graph Theory*, pages 213–248. Springer, 2013. doi:10.1007/978-1-4614-0110-0_12.

11 Daniel Gonçalves, Lucas Isenmann, and Claire Pennarun. Planar graphs as L-intersection or L-contact graphs. In *Discrete Algorithms (SODA'18)*, pages 172–184. SIAM, 2018. doi:10.1137/1.9781611975031.12.

12 Petr Hliněný. Classes and recognition of curve contact graphs. *Journal of Combinatorial Theory, Series B*, 74(1):87–103, 1998. doi:10.1006/jctb.1998.1846.

13 Petr Hliněný and Jan Kratochvíl. Representing graphs by disks and balls (a survey of recognition-complexity results). *Discrete Mathematics*, 229(1):101–124, 2001. doi:10.1016/S0012-365X(00)00204-1.

14 Boris Klemz, Martin Nöllenburg, and Roman Prutkin. Recognizing weighted disk contact graphs. In *Graph Drawing and Network Visualization (GD'15)*, volume 9411 of *LNCS*, pages 433–446. Springer International Publishing, 2015. doi:10.1007/978-3-319-27261-0_36.

15 Paul Koebe. Kontaktprobleme der konformen Abbildung. *Ber. Sächs. Akad. Wiss. Leipzig, Math.-Phys. Klasse*, 88:141–164, 1936.

On the Realizability of Free Space Diagrams*

Maike Buchin[1], Leonie Ryvkin[2], and Carola Wenk†[3]

1 **Department of Mathematics, Ruhr University Bochum, Germany**
`maike.buchin@rub.de`
2 **Department of Mathematics, Ruhr University Bochum, Germany**
`leonie.ryvkin@rub.de`
3 **Department of Computer Science, Tulane University, Louisiana, USA**
`cwenk@tulane.edu`

Abstract

The free space diagram is a popular tool to compute the well-known Fréchet distance. Here we ask, whether given patterns in the free space diagram can be *realized* by two polygonal curves in the plane. We show how to compute realizing curves for certain input instances. However, for arbitrary instances we show that maximizing the number of realizing cells is NP-hard.

1 Introduction

The Fréchet distance is an important distance measure for curves that is considered in many applications. For two curves, $P, Q\colon I \to \mathbb{R}^2$, where $I \subseteq \mathbb{R}^2$, their *Fréchet distance* is $\delta_{\mathrm{F}}(P,Q) = \inf_\sigma \max_{t\in 0,1} \|P(t) - Q(\sigma(t))\|$, where the reparameterization σ ranges over all orientation-preserving homeomorphisms. Intuitively, suppose a woman is walking her dog, each traversing one of the curves. Then the Fréchet distance is the length of the shortest leash allowing both to traverse their entire curves continuously, choosing their speed independently.

A popular tool for computing, and deciding, the $\delta_{\mathrm{F}}(P,Q)$ is the *free space diagram* $D_\varepsilon(P,Q)$, which is the cross-product $I \times I$ of the parameter spaces of the curves partitioned into *free space* and its complement. For a given $\varepsilon > 0$, *free space* is defined as $F_\varepsilon(P,Q) = \{(r,t) : \|P(r) - Q(t)\| \leq \varepsilon\}$. Then $\delta_{\mathrm{F}}(P,Q) \leq \varepsilon$ iff there exists a non-decreasing path in $F_\varepsilon(P,Q)$ that covers the parameter spaces of both curves. For polygonal curves P, Q of n, m segments, $D_\varepsilon(P,Q)$ can be subdivided into an $n \times m$ grid, where each *cell* C_{ij} corresponds to a pair of segments $s_i^P \subseteq P$, $s_j^Q \subseteq Q$, for $1 \leq i \leq n$, $1 \leq j \leq m$. Then it takes $O(mn)$ time using dynamic programming to find a path in free space to decide whether $\delta_{\mathrm{F}}(P,Q) \leq \varepsilon$ [2].

For different applications, many variants of the Fréchet distance have been developed. These are typically computed using the free space diagram, and thus its complexity impacts the runtimes. While the Fréchet distance cannot be computed in subquadratic time unless SETH fails [4], there are faster algorithms for computing the free space diagram for special curve classes [3, 6], which exploit a special structure of free space. These bounds consider the worst-case complexity of the free space diagram, but not every diagram can be realized by a pair of curves, see Figure 1. A few variants of the Fréchet distance have been proven to be NP-hard to decide [1, 5], some of which build certain free space diagrams for the reduction.

Problem statement. Given a free space diagram D_ε, do there exist curves P, Q such that $D_\varepsilon = D_\varepsilon(P,Q)$? And if so, can we construct P, Q? Understanding this *free space realizability problem* will give structural insights into free space and the computation of the Fréchet distance, in particular for special curve classes.

* We thank Majid Mirzanezhad for fruitful discussions.
† Partially supported by NSF grant CCF 1637576.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

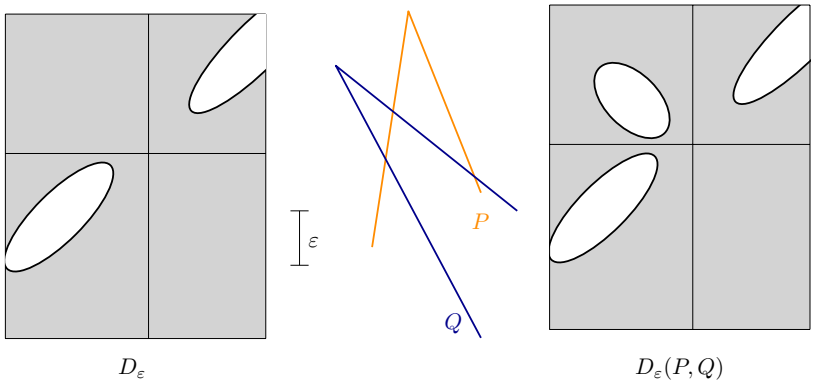


Figure 1 A given free space diagram D_ε which cannot be realized by two curves in $\mathbb{R}^2$. But $D_\varepsilon(P,Q)$ has all but one cell in common with D_ε.

More notation. If a *polygonal curve* $P\colon I \to \mathbb{R}^2$ is defined by points $p_0, \ldots, p_n \in \mathbb{R}^2$, then let $t_i \in I$ such that $P(t_i) = p_i$, and $s_i^P = \overline{p_{i-1}p_i}$ be the line segment connecting p_i and p_j. The *arclength parameterization* of P linearly interpolates between consecutive points, i.e., $P(i+\lambda) = (1-\lambda)p_i + \lambda p_{i+1}$ for $t_i = i/\ell$ and $\lambda \in [0,1]$, for $i = 0, \ldots, n-1$, where ℓ is the *arclength* of P, and $I = [0, \ell]$. A *free space component* is $c_{ij} = C_{ij} \cap F_\varepsilon = F_\varepsilon(s_i^P, s_j^Q)$. Points $(r,t) \in F_\varepsilon$ are called *white*, and points $(r',t') \notin F_\varepsilon$ *black*. A cell C_{ij} is *empty* (or *black*) if $C_{ij} \cap F_\varepsilon = \emptyset$, *full* (or *white*) if $C_{ij} \cap F_\varepsilon = C_{ij}$, and *partially full* if $\emptyset \neq C_{ij} \cap F_\varepsilon \neq C_{ij}$. An empty cell results from two segments that do not intersect each others' ε-neighborhoods; a full cell stems from segments that are completely contained in each others' ε-neighborhoods. In the partially full case, the free space $F_\varepsilon(s_i^P, s_j^Q)$ is an ellipse cropped at the cell boundaries. A *free space strip* denotes a number of cells that form a connected portion of a single row or column in the free space diagram.

Our contributions. We study the free space realizability problem, assuming arc-length parameterizations of the curves, which means that cell boundaries, i.e., segment lengths, are not uniform. We assume we are provided with a detailed description of D_ε, including ε, the lengths of cell boundaries, and exact locations of sufficiently many points on the boundary curves of all c_{ij}. In Section 2 we observe several patterns for which we can easily compute realizing curves. In Section 3 we compute realizing curves for certain input instances. In Section 4 we show that maximizing the number of realizing cells for arbitrary instances is NP-hard. We continue to work on realizing more general free space diagrams.

2 Observations

The following proposition characterizes the shape and position of a free space component within a cell. Its proof and further descriptions will be given in [9].

▶ **Proposition 2.1.** *A component c_{ij} equals the intersection of an ellipse with the cell C_{ij}, as shown in [2]. The major axis of the ellipse forms a $\pm 45°$ angle with the cell boundaries [8].*

We briefly sketch the proof of the following

▶ **Lemma 2.2.** *Given ε, a partially full free space cell C_{ij}, and the equation of the ellipse's boundary curve within the cell, we can compute the corresponding segments' relative placement.*

Consider cell C_{ij}. From the boundary curve of the ellipse E_{ij}, where $c_{ij} = E_{ij} \cap C_{ij}$ is the component within our cell, we can easily derive its center and extremal points. Assuming

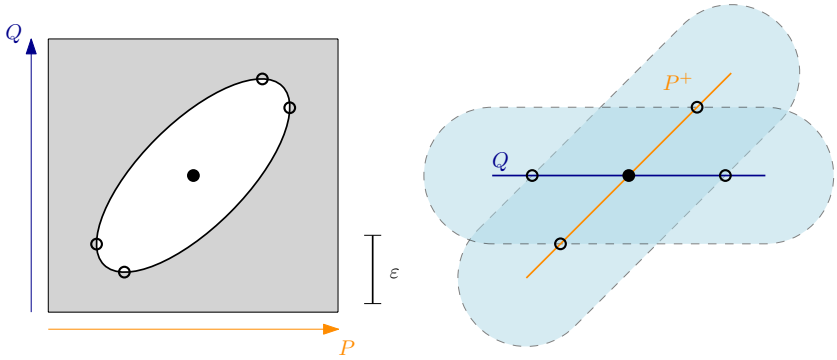


Figure 2 A free space diagram consisting of one cell with marked extremal points and center, and one corresponding realization. We fix Q and place P choosing a positive enclosed angle.

that the cell's boundaries coincide with x- and y-axis of $\mathbb{R}^2$, and that the bottom left corner has coordinates $(0,0)$, the ellipse's center (r_i, t_j) holds the intersection point of the two segments s_i^P, s_j^Q. The length of s_i^P from its starting point to the intersection with s_j^Q equals r_i, analogously for s_j^Q and t_j. Extremal points encode intersections of one segment with the ε-neighborhood of the other segment, as shown in Figure 2.

Free space strips. Recall that a strip is a (partial) row or column in the free space diagram, hence it corresponds to one segment of curve P, and a number of connected segments of the second curve Q, or vice versa. Given the strip, we want to either give the positions of all segments, or decide that the strip is not realizable. A completely full or completely empty strip does not provide enough information to place the corresponding segments, in contrast to partially full cells. We call the consecutive empty or full cells in between two partially full cells within a strip a *gap*. If the cells in a gap are empty (resp., full), we call this a *black (resp., white) gap*. The *size* of a gap is the number of cells.

▶ **Lemma 2.3.** *For a given free space strip with maximum gap size 2, we can determine realizability* and *compute valid placements of the corresponding segments.*

For larger gaps, i.e., more than two empty/full cells in between two partially full ones, the number of possible placements for a segment is not discrete in general, thus constructing curves from such strips becomes computationally harder. The following lemma and figure illustrate cases in which we are able to compute placement options for empty or full cells using placement options of cells in their vicinity.

▶ **Lemma 2.4.** *It is possible to derive the pairwise distances between endpoints of two segments for full or empty cells in these cases:*

1. *If we know the placements of segments $s_\ell^P, \dots, s_r^P$ with respect to s_j^Q (horizontal strip $C_{\ell j}, \dots, C_{rj}$), as well as the placements of $s_b^Q, \dots, s_t^Q$ with respect to s_i^P (vertical strip $C_{ib}, \dots, C_{it}$), where $\ell \le i \le r$, we can compute the placements for $s_\ell^P, \dots, s_r^P$ with respect to each segment $s_b^Q, \dots, s_t^Q$.*
2. *If we know the placements of segments $s_i^P, \dots, s_k^P$ with respect to s_b^Q, and of $s_\ell^P, \dots, s_r^P$ with respect to s_t^Q, where $\ell \le k$ (two strips within parallel rows/columns $C_{ib}, \dots, C_{kb}$ and $C_{\ell t}, \dots, C_{rt}$ with overlapping projections), we can compute the placements of $s_{k+1}^P, \dots, s_r^P$ with respect to s_b^Q, and of $s_i^P, \dots, s_{\ell-1}^P$ with respect to s_t^Q.*
3. *If we know the placements corresponding to cells sharing a boundary vertex, C_{ij} and C_{i+1j+1}, we can also compute the placement of the segments corresponding to their neighboring cells C_{ij+1} and C_{i+1j}.*

4. *If we know the placements of segments s_i^P with respect to s_j^Q, as well as s_{i+2}^P with respect to s_{j+1}^Q (cells C_{ij} and C_{i+2j+1}), we can also compute the placements of s_i^P and s_{i+1}^P with respect to s_{j+1}^Q, as well as of s_{i+1}^P and s_{i+2}^P with respect to s_j^Q.*

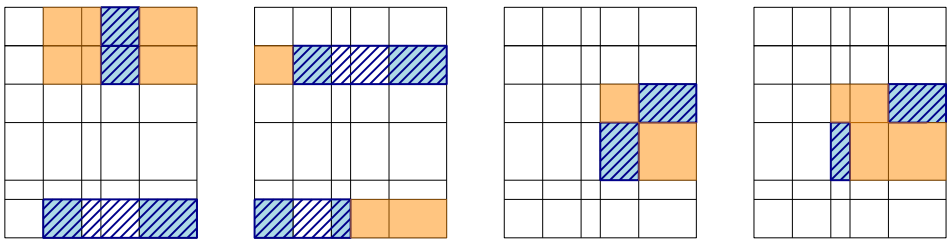


Figure 3 The grid represents a free space diagram, light blue areas depict partially full cells, and cells with blue stripes correspond to segments whose relative positions have been computed. We extend the computed information to the orange cells.

3 Polynomial Time Realizability Algorithm

Given a free space diagram, we decide realizability and compute realizing curves as follows. First, we compute the *placement graph* containing information on the relative placements of segments. Recall that the numbers and lengths of segments are part of the input.

We define the placement graph as follows. Let $G = (V, E)$ be a graph, where the vertices $V = \left\{v_1^P, \ldots, v_n^P, v_1^Q, \ldots, v_m^Q\right\}$ correspond to segments of P and Q. We initialize $E = \emptyset$. An edge $\left\{v_i^P, v_j^Q\right\}$ is later inserted for every cell C_{ij} we compute information on, i.e., where we obtain the relative position of s_i^P with respect to s_j^Q (or vice versa) by applying Lemmas 2.2, 2.3 and 2.4. We also add edges $\left\{v_i^P, v_{i+1}^P\right\}$ or $\left\{v_j^Q, v_{j+1}^Q\right\}$ as soon as we know the enclosed angle between consecutive segments (and hence their relative positions). This occurs, e.g., for neighboring partially full cells. We store these relative positions (enclosed angle for consecutive segments, distances between all pairs of endpoints for segments of different curves) with each edge.

To ensure polynomial runtime, we need that sufficiently many cells are partially full and that they are "well-distributed" throughout the free space diagram. That is, we ask to compute the relative placement of each segment with respect to one fixed segment, without having to deal with gaps of size two or larger. Smaller gaps as well as partially full cells that share a common corner (Case 3 of Figure 3) result in at most two symmetric placement options, which we can handle efficiently. However, note that we can handle cases with such gaps as long as a sufficient amount of information is provided, see Figure 4.

The overall idea is as follows: as described above, we initialize the placement graph $G = G_{\text{poly}}$ with vertices for each segment of the two curves. Whenever we obtain information on the relative placement of one segment with respect to another one, we add an edge between their corresponding vertices and store the placement of the segments with it. This way, we can later access the relative position of a segment and place it accordingly. More detailed descriptions and pseudocode will be presented in [9]. Then we visit all cells "outside-in", i.e., we first consider cells C_{1j}, C_{nj}, C_{i1} and C_{im} for $i = 1, \ldots, n, j = 1, \ldots, m$, then cells C_{2j}, C_{n-1j}, C_{i2} and C_{im-1}, and so on. We proceed cell per cell, marking each cell we are able

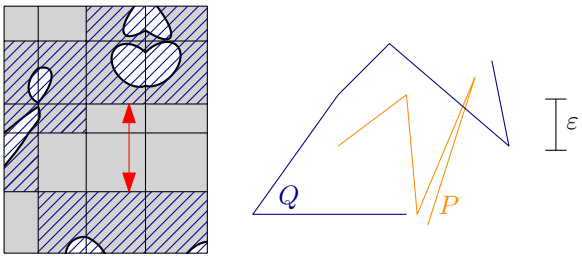


Figure 4 A free space providing sufficiently many well-distributed partially full cells (blue striped) to piece together curves P and Q, even though there is a gap of size 2 (marked in red).

to compute information on as "visited". We start by adding edges for partially full cells and compute relative placements for these (using Lemma 2.2). We mark the considered cell as visited and examine its surrounding cells: First, we check whether a neighboring cell (a cell sharing a boundary edge) is also visited and add an edge between the consecutive segments' vertices; then, we check for visited diagonal cells, i.e., cells that share a boundary vertex, and add the corresponding edges, as well. Finally, we consider gaps of size one: for each cell, we check whether it has an empty or full neighboring cell, which borders another visited cell within the same strip. In that case we compute the position of the "missing" segment using Lemma 2.3 and add the corresponding edges.

When after processing the free space diagram and adding edges to G_{poly} in this way, there are two vertices $v_i^P, v_j^Q \in G_{\text{poly}}$ (i.e., segments s_i^P, s_j^Q), that are connected to all vertices (and thus in particular G_{poly} is connected), we can build realizing curves for the free space diagram, or decide that it is not realizable. A formal proof of this claim will be given in [9], and pseudocode of our algorithmic approaches will also be presented in [9]. We obtain

▶ **Theorem 3.1.** *For a free space diagram D_ε with sufficiently many well-distributed partially full cells, our algorithm decides whether it is realizable through curves and outputs such curves if possible. The algorithm runs in time $\mathcal{O}((n+m)^2 + nm\log(nm))$, for D_ε of size $\mathcal{O}(nm)$, provided only a constant number k of endpoints p_i, q_j coincide.*

As stated, our algorithm only works in case sufficiently many partially full cells are well-distributed throughout the given diagram. This is, e.g., the case if all cells are either partially full themselves, or border at least two partially full cells. In Figure 3, however, we show more situations in which we are able to derive placement options. These cases lead to an exponential runtime algorithm. This algorithmic extension will be presented in [9].

Note that characterizing whether the realizability of a given diagram is decidable in polynomial or exponential time (or not at all, for our algorithms) requires analyzing the diagram and all possibilities to compute information on empty or full cells, which seems to be as computationally expensive as running our algorthims in the first place.

4 Maximizing the Number of Realized Cells is NP-Hard

Our algorithm can only decide realizability in polynomial time for certain instances. Next, we show that in fact maximizing the number of realized cells is NP-hard for arbitrary instances. For this we reduce from (the decision version of) **Monotone MAX-1-in-2-SAT**, i.e., given a 2-SAT formula without negations, the goal is to maximize the number of clauses where exactly one variable is set to `true`. For completeness, we prove the NP-hardness for this problem by a reduction from the MAX-CUT problem [7] in [9].

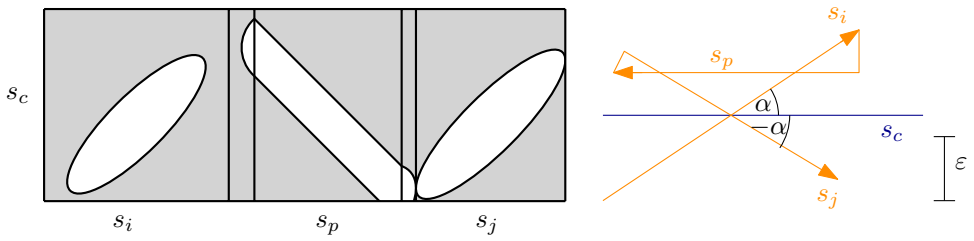


Figure 5 A strip representing a monotone 1-in-2-clause and the curves realizing them. The segment s_i has a positive and s_j has a negative angle, indicating that the i-th variable should be assigned to `true` and the j-th variable to `false` to fulfill the corresponding 1-in-2-clause.

▶ **Theorem 4.1.** *Deciding if two polygonal curves exist that realize at least k cells of a given free space diagram is NP-hard.*

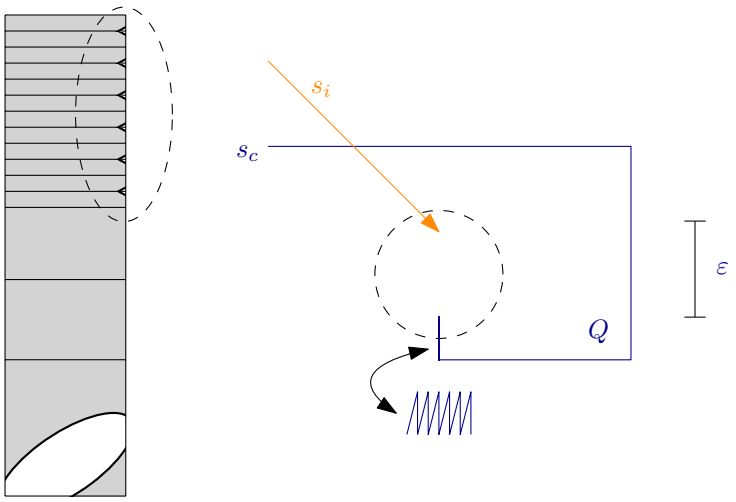


Figure 6 A strip corresponding to segment $s_i \in P$, the (prolonged) clause-segment, and control gadget (in dashed ellipse). To the right, realizing curves are shown with the control gadget (in dashed circle) consisting of multiple segments drawn on top of each other (shown perturbed below).

Proof. This is a rough sketch of proof, the detailed proof will be given in [9]. We reduce from monotone MAX 1-in-2-SAT: for a given 2-SAT formula of n variables and m clauses, we construct a free space diagram. In this diagram, a maximizing variable assignment corresponds to placing curves that realize the maximum number of free space cells.

First recall that for a given fixed segment, placing another segment at angle α results in the same free space cell as placing the segment at angle $-\alpha$. We use this to model setting a variable to `true` or `false`. Based on this, we construct a clause-gadget and a control-gadget, the latter of which ensures that segments representing variables are placed consistently.

The clause gadget is shown in Figure 5. It consists of 5 cells; the outer two correspond to the two variables, and the inner three can be realized exactly if exactly one of the variables is set to `true`, i.e., the two segments are placed one at positive and one at a negative angle.

To ensure that variable segments are placed consistently, we place control gadgets over each copy of a variable segment, see Figure 6. The control gadgets are sufficiently large, which enforces realizing these cells, and hence assigning variables consistently. By placing different variables slightly differently, their control gadgets do not interfere with other variables.

Figure 7 shows a (tiny) complete construction. In the second from bottom row the clause gadgets are placed next to each other. In between two clause gadgets we add a partially full cell in the bottom row, which corresponds to visiting a "neutralizing" segment inbetween. And control gadgets are added above each occurrence of a variable (here only two). ◀

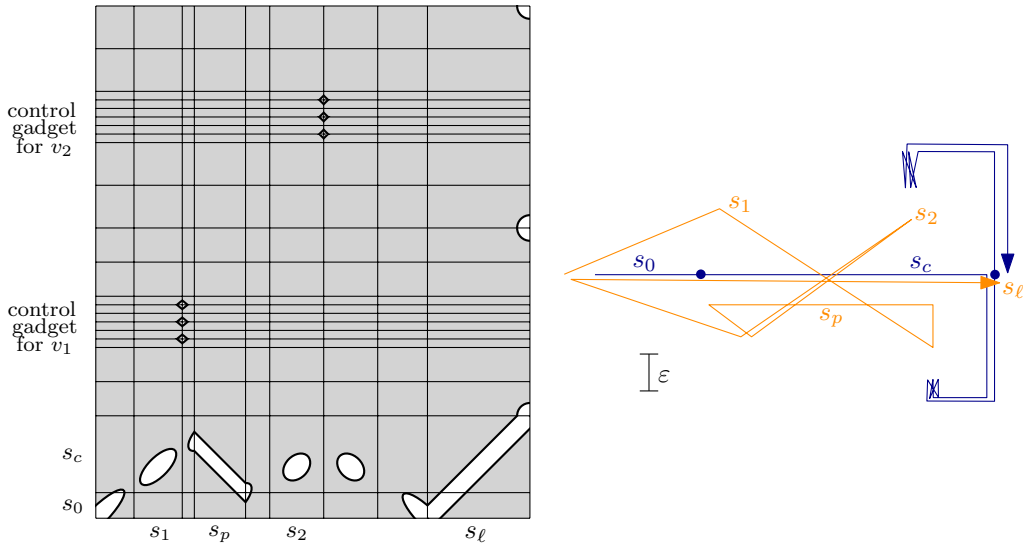


Figure 7 The free space diagram for the clause $(v_1 \vee v_2)$ with clause and both control gadgets, each featuring three spikes. On the right hand side a perturbed drawing of realizing curves, where s_1 has a positive and s_2 a negative angle to s_c, i.e., we would set v_1 to `true` and v_2 to `false`.

References

1 Hugo Alves Akitaya, Maike Buchin, Leonie Ryvkin, and Jérôme Urhausen. The k-Fréchet distance: How to walk your dog while teleporting. In *30th International Symposium on Algorithms and Computation*, pages 50:1–50:15, 2019.

2 Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *Internat. J. Comput. Geom. Appl.*, 5(1-2):75–91, 1995.

3 Helmut Alt, Christian Knauer, and Carola Wenk. Comparison of distance measures for planar curves. *Algorithmica*, 38(1):45–58, 2004.

4 Karl Bringmann. Why walking the dog takes time: Fréchet distance has no strongly subquadratic algorithms unless SETH fails. In *55th Annual Symposium on Foundations of Computer Science*, pages 661–670, 2014.

5 Maike Buchin, Anne Driemel, and Bettina Speckmann. Computing the Fréchet distance with shortcuts is NP-hard. In *Proceedings of the Thirtieth Annual Symposium on Computational Geometry*, page 367–376, 2014.

6 Anne Driemel, Sariel Har-Peled, and Carola Wenk. Approximating the Fréchet distance for realistic curves in near linear time. *Discrete Comput. Geom.*, 48(1):94–127, 2012.

7 Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43(3):425 – 440, 1991.

8 Günter Rote. Lexicographic Fréchet matchings. In *30th European Workshop on Computational Geometry*, 2014.

9 Leonie Ryvkin. *Novel Distance Measures for Polygonal Curves : Bridging Between Hausdorff and Fréchet Distance*. PhD thesis, Ruhr University Bochum, (to appear) May 2021.

Accelerating Amoebots via Reconfigurable Circuits*

Michael Feldmann[1], Andreas Padalkin[2], Christian Scheideler[3], and Shlomi Dolev[4]

1 **Paderborn University**
michael.feldmann@upb.de
2 **Paderborn University**
andreas.padalkin@upb.de
3 **Paderborn University**
scheideler@upb.de
4 **Ben-Gurion University of the Negev**
dolev@cs.bgu.ac.il

Abstract

We consider an extension to the geometric amoebot model that allow particles to form a constant amount of *circuits*. Given a connected particle structure, a circuit is a subgraph formed by the particles that permits the instant transmission of signals. We show that such an extension allows for significant faster solutions to a variety of problems related to programmable matter. More specifically, we provide algorithms for leader election and compass alignment. Leader election can be solved in $\Theta(\log n)$ rounds, w.h.p., and compass alignment can be solved in $O(\log n)$ rounds, w.h.p.

1 Introduction

Programmable matter is a physical substance consisting of tiny, homogeneous robots (also called *particles*) that is able to dynamically change its physical properties like shape or density. Such a substance can be deployed, for example, to build any objects imaginable or for the detection of cancer cells through injection into the human body. Programmable matter has been envisioned for 30 years [23] and is yet still to be realized in practice. However, theoretical investigation on various models (such as the self-assembly model [21], the nubot model [24] or the geometric amoebot model [8]) have already been started and is still continuing in the distributed computing community.

Our investigation on programmable matter is made in the geometric amoebot model where the locomotions of the particles are inspired by amoeba. However, the model only supports movements by single particles at a time. We aim to accelerate algorithms for this model by synchronizing the locomotions of several particles, which leads to a faster reorganization of whole parts of particle structures. For this purpose, we introduce an extension to the geometric amoebot model. Each particle is allowed to create a constant amount of *circuits* with a subset of particles from the network. A circuit formed by particles allows for the instantaneous transmission of primitive signals.

In spite of our prime goal, we start our investigation of the new possibilities on fundamental algorithms in a stationary setting. More precisely, we study algorithms for *leader election* and *compass alignment*. Leader election has turned out as a crucial primitive to break symmetries

* This work has been supported by the DFG Project SFB 901 (On-The-Fly Computing) and the DFG Project SCHE 1592/6-1 (PROGMATTER).

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

in various geometric problems, e.g., shape formation (see for example [10, 17]). Compass alignment ensures that the synchronized movements of particles are coordinated into the same direction. We show that we are able to achieve a significant improvement on previous results with the help of the circuits. Altogether, this paper will lay the foundation for an upcoming examination of problems requiring locomotion, e.g., *shape formation* and *object coating*.

The full version of this paper will contain solutions for the shape recognition problem.

1.1 Model

We introduce the extension of the *geometric amoebot model* from [8], which we describe in the following. In this model, a set of n uniform particles is placed on the infinite regular triangular grid graph $G_{eqt} = (V, E)$ (see Figure 1a). The particles are anonymous, randomized finite state machines. Each particle occupies a single node[1] and each node is occupied by at most one particle.

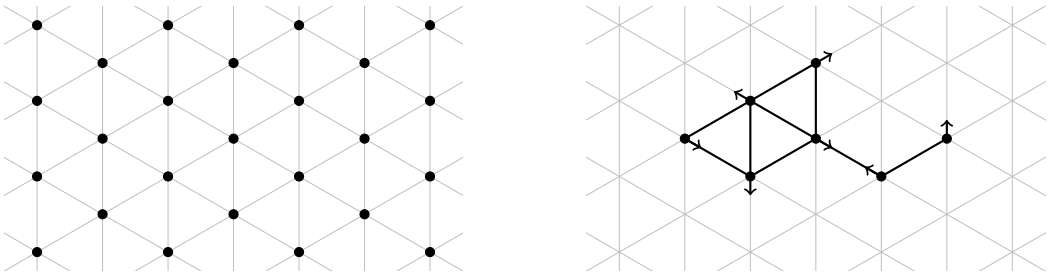


Figure 1 (a) A section of G_{eqt}. (b) A connected particle structure. The nodes are the particles and the edges are the bonds. A small arrow indicates the orientation of each particle.

A bond is formed between particles occupying adjacent nodes. These particles are said to be neighbors. Neighbors are able to communicate through a locally shared constant-size memory. Furthermore, particles assign a locally unique label to each edge incident to their occupied nodes. We assume that all particles share a common chirality. However, the particles do not have to agree on a common orientation.

Let the *particle structure* $S \subseteq V$ be the set of nodes occupied by the particles (see Figure 1b). It is connected iff G_S is connected where $G_S = G_{eqt}|_S$ is the graph induced by S. We expect that the structure is connected[2].

In our model extension, *pins* are added to each bond between two particles. The number of pins per bond is bounded by a constant c. A unique identifier is assigned to each pin of a bond. These are known to both particles such that the pins can be distinguished. Particles can connect the pins of their bonds arbitrarily.

Let P be the set of all pins and let C be the set of all connections between pins. Then, we define a *circuit* as a connected component of graph $G_C = (P, C)$ (see Figure 2). A particle is part of a circuit iff the circuit contains at least one pin on one of its bonds.

[1] Actually, particles are allowed to occupy either a single node or a pair of adjacent nodes. W.l.o.g., we can assume the former for all particles. Otherwise, we let the latter simulate 2 individual particles.

[2] Particles have the ability to move by expansions and contractions. Since the presented algorithms work entirely stationary, we omit further details and refer to [8]. The particles have to maintain a connected structure at all times.

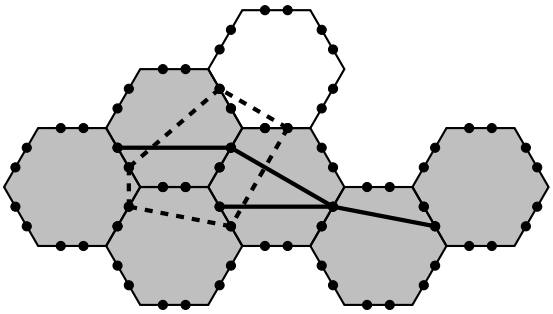


Figure 2 The inner perspective of an particle structure for $c = 2$. The hexagons represent the particles. Each side is a bond. The pins P are indicated by nodes. The connections C are indicated by solid and dashed edges. Both the solid and dashed edges indicate a circuit. The particles connected by the solid circuit are highlighted in gray.

Each particle can send a primitive signal (a *beep*) through its circuits that is received by all particles of the same circuit. The particles receive a beep if at least one particle sends a beep on the circuit but the particles do neither know the origin of the signal nor the number of origins.

Particles operate in *look-compute-act* cycles: In the look phase the particle gathers information. It reads its local and shared memories. Moreover, it may receive beeps on its circuits. In the compute phase it performs some calculations and may update its local and shared memory. In the act phase it may reconfigure the connections of its pins and send a signal through an arbitrary number of its circuits. The signals are received during the next look phase. Note that we assume a constant transmission time on the circuits, regardless of the size of the circuit. This assumption is adopted from related *reconfigurable network models* (see Section 1.3).

We assume that initially, all particles are connected to a single circuit, which we call the *global circuit*, i.e., each particle connects a predefined pin of each bond together. Furthermore, we assume that the particle structure progresses in synchronized rounds. Initially, the particles may not be synchronized. But we do assume that they have a common sense of a time unit, so they will synchronize each other at the first activation of the global circuit. We measure the complexity of an algorithm by the number of synchronized rounds required after the first activation of the global circuit.

1.2 Problem Statement and Our Contribution

We consider the following two fundamental problems, given a connected particle structure S.

First, we study the *leader election problem*. The particles have to agree on exactly one single particle, which becomes the leader. Since particles only have constant storage capacity and due to the absence of node identifiers, leader election is a non-trivial problem. We propose a protocol, which requires $\Theta(\log n)$ rounds, w.h.p.[3] This is a significant improvement on the runtime of previous algorithms (see Section 1.3), which have at least linear complexity.

Second, we examine the *compass alignment problem*. The particles do not agree on a common orientation. Thus, the goal of the problem is to align the compasses of all particles

[3] An event holds with high probability (w.h.p.) if it holds with probability at least $1 - 1/n^c$ where the constant c can be made arbitrarily large.

globally. A compass alignment is essential to coordinate synchronized movements. Our algorithm requires $O(\log n)$ rounds, w.h.p.

1.3 Related Work

Reconfigurable circuits have proven their value in various *reconfigurable network models*, e.g., polymorphic-torus networks [15, 16], meshes with reconfigurable bus [16, 18] and bus automata [19, 22]. Further examples can be found in [2, 16]. We will discuss the generalized model stated in [2]. Usually, a square mesh is utilized as network topology. A processor with a switch is placed on each node. For the particular model, the computational power of the processors may vary. The switches control the connectivity of the incident edges. The possible connections may be restricted in different variants. A connected component is called a bus.

Each processor connected to a bus may transmit a message on the bus, which is received by all other processors of the same bus. Contrary to our model, non-primitive messages are possible. Depending on the specific model, collisions either are detected or go undetected if more than one processor has transmitted a message on a bus. In the former case the massages are assumed to be destroyed. In the latter case, for example, a logical OR may be applied to the messages. Our model corresponds to the latter.

The leader election problem is an extensively researched problem within the geometric amoebot model [1, 4, 7, 12, 13, 14, 17]. To our knowledge there are no publications regarding compass alignment within the amoebot model. Further publications include shape formation [6, 9, 10, 12, 17], gathering [3], and object coating [5, 11].

2 Leader Election

In this section we give an efficient solution for the leader election problem. The underlying idea has been analyzed before, for example in [20]. Our protocol works in two phases. First, we reduce the number of particles that are being considered for the leader from n to $O(\log n)$. The second phase then elects a leader among the remaining $O(\log n)$ candidates, w.h.p.

Throughout the protocol, we maintain a set $C_1 \subseteq S$ of particles that are considered as *candidates* for the leader. In the first phase we perform a *tournament* on the particles that are still candidates to become the leader. Initially, each particle is a candidate, i.e., $C_1 = S$.

A single *iteration* of the tournament works as follows. Each candidate $u \in C_1$ tosses a coin that is either *HEADS* or *TAILS* and stores the result in a variable $u.c_1$. Now consider two subsequent rounds r_1, r_2. In round r_1 all candidates $u \in C_1$ with $u.c_1 = \mathit{HEADS}$ send a signal through the global circuit and in r_2 all candidates $u \in C_1$ with $u.c_1 = \mathit{TAILS}$ send a signal through the global circuit. As signals are received instantly by all particles, each particle is able to check if there is a candidate that beeped in r_1 and one that beeped in r_2. If at least one candidate beeped in r_1, all candidates that beeped in r_2 are out of contention for being the leader (therefore, the set C_1 gets updated). We continue performing iterations of the tournament until we reach an iteration where in either round r_1 or in r_2 no candidate beeped. This finishes the first phase of our protocol and takes $\Theta(\log n)$ rounds, w.h.p.

In the second phase we continue the tournament from the first phase on the remaining $O(\log n)$ candidates in C_1 until there is only one single candidate left. Unfortunately, we cannot check efficiently when this is the case because we cannot count the number of particles that have sent a signal in a single round through the global circuit. Therefore, we aim to continue the tournament from the first phase for another $\Theta(\log n)$ rounds. As particles cannot count to $\Theta(\log n)$ due to their constant-sized storage, we just perform a second tournament

on the set C_2, initially set to S, in parallel to the tournament on the set C_1. One iteration of the second phase therefore consists of 4 rounds – 2 rounds for the tournament on C_1 and 2 rounds for the tournament on C_2. The second tournament terminates once the tournament on C_2 has finished. By repeating the second tournament κ times for a constant $\kappa \geq 3$, we can guarantee that w.h.p., only one single node remains as a candidate.

▶ **Theorem 2.1.** *A particle structure elects a leader within* $\Theta(\log n)$ *rounds, w.h.p.*

3 Compass Alignment

In this section, we consider another basic task: the alignment of the compasses of all particles. Our protocol divides the particle structure into regions of homogeneous compass alignments and fuses these iteratively into a single region by adjusting the alignments of regions.

The regions are defined by the connected components of graph $G_R = (S, A)$ where $A = \{\{u, v\} \in E \mid u, v \text{ aligned}\}$. Let $\mathcal{R}_t$ denote the set of all regions after the t-th loop and $\mathcal{R}_0$ denote the set of all regions at the beginning. Moreover, we call region R' a neighbor of region R if there exists $u \in R$ and $v \in R'$ such that these are adjacent in G_S. The neighborhood of R is given by $\mathrm{N}(R)$.

In a nutshell, loop t proceeds as follows: First, we determine set $\mathcal{R}_{t-1}$ and connect the particles of each region by a regional circuit. Each region $R \in \mathcal{R}_{t-1}$ attempts to toss a coin. Thereupon, each region that has tossed *TAILS* fuses into a neighbor region that has not tossed *TAILS* by adjusting its compass. The new regions are determined at the beginning of the next loop. Further fusions may occur beyond these adjustments (see Figure 3).

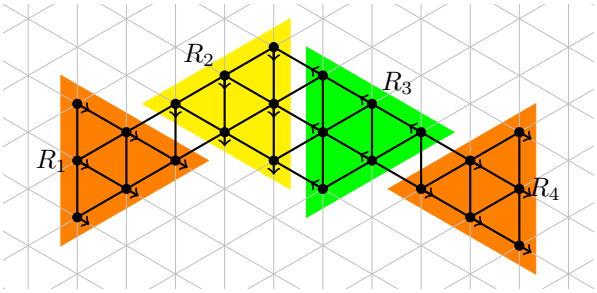


Figure 3 Region R_1 and R_4 have the same orientation. Region R_2 adjusts its compass to region R_1. Region R_3 adjusts its compass to region R_4. All 4 regions fuse together.

In order to toss a coin, we use a similar approach as for the leader election. Each region contains a non-empty set of candidates. Exactly like the candidates in the leader election protocol, each of these tosses a coin and beeps in the respective round. A coin toss is successful if all coin tosses of the candidates coincide. Otherwise, the candidacies of all candidates that have tossed *TAILS* are revoked.

Suppose that region R has tossed *TAILS*. We now describe how region R fuses into another region. Let $\mathrm{N}'(R)$ denote all neighbors of R that have not tossed *TAILS*. These regions do not change their orientation in the current loop. We define $\mathrm{offset}(R, R')$ as the minimal number of clockwise rotations (by 60°) necessary to adjust the particles of R to the compass of R'. Note that $\mathrm{offset}(R, R') \in [1, 5]$.

Particles adjacent to a neighboring region can obtain information from these neighboring regions (by reading from the shared memory). In particular, they know about the result of the coin tosses and about the offset. Consider the particles in region R that are adjacent to

a region that has not tossed *TAILS*. We divide these particles into 5 subsets accordingly to the offsets. Note that the subsets are not necessarily disjoint. In more detail, let

$$\forall i \in [1,5] : B_i = \{u \in R \mid \exists R' \in \mathrm{N}'(R) : \mathrm{offset}(R, R') = i \wedge \exists v \in R' : u, v \text{ adjacent in } G_S\}.$$

Note that $B_i \neq \emptyset$ implies that region R can fuse into another region by rotating its particles i times in clockwise direction. Similar to the coin tosses, we consider 5 subsequent rounds $r_1, \ldots, r_5$. Each particle $u \in B_i$ beeps on the regional circuit in round r_i. The region R may pick the first possible offset.

Finally, we have to check if there is still more than one region left. Any particle that has an unaligned neighbor beeps continuously on the global circuit. The algorithm terminates once a round is reached where the global circuit has not been activated.

▶ **Theorem 3.1.** *The compasses of all particles are aligned after $O(\log n)$ rounds, w.h.p.*

References

1 Rida A. Bazzi and Joseph L. Briones. Stationary and deterministic leader election in self-organizing particle systems. In Mohsen Ghaffari, Mikhail Nesterenko, Sébastien Tixeuil, Sara Tucci, and Yukiko Yamauchi, editors, *Stabilization, Safety, and Security of Distributed Systems - 21st International Symposium, SSS 2019, Pisa, Italy, October 22-25, 2019, Proceedings*, volume 11914 of *Lecture Notes in Computer Science*, pages 22–37. Springer, 2019. doi:10.1007/978-3-030-34992-9_3.

2 Yosi Ben-Asher, Klaus-Jörn Lange, David Peleg, and Assaf Schuster. The complexity of reconfiguring network models. *Inf. Comput.*, 121(1):41–58, 1995. doi:10.1006/inco.1995.1122.

3 Sarah Cannon, Joshua J. Daymude, Dana Randall, and Andréa W. Richa. A markov chain algorithm for compression in self-organizing particle systems. In George Giakkoupis, editor, *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing, PODC 2016, Chicago, IL, USA, July 25-28, 2016*, pages 279–288. ACM, 2016. doi:10.1145/2933057.2933107.

4 Gianlorenzo D'Angelo, Mattia D'Emidio, Shantanu Das, Alfredo Navarra, and Giuseppe Prencipe. Asynchronous silent programmable matter achieves leader election and compaction. *IEEE Access*, 8:207619–207634, 2020. doi:10.1109/ACCESS.2020.3038174.

5 Joshua J. Daymude, Zahra Derakhshandeh, Robert Gmyr, Alexandra Porter, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. On the runtime of universal coating for programmable matter. *Nat. Comput.*, 17(1):81–96, 2018. doi:10.1007/s11047-017-9658-6.

6 Joshua J. Daymude, Robert Gmyr, Kristian Hinnenthal, Irina Kostitsyna, Christian Scheideler, and Andréa W. Richa. Convex hull formation for programmable matter. In Nandini Mukherjee and Sriram V. Pemmaraju, editors, *ICDCN 2020: 21st International Conference on Distributed Computing and Networking, Kolkata, India, January 4-7, 2020*, pages 2:1–2:10. ACM, 2020. doi:10.1145/3369740.3372916.

7 Joshua J. Daymude, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. Improved leader election for self-organizing programmable matter. In Antonio Fernández Anta, Tomasz Jurdzinski, Miguel A. Mosteiro, and Yanyong Zhang, editors, *Algorithms for Sensor Systems - 13th International Symposium on Algorithms and Experiments for Wireless Sensor Networks, ALGOSENSORS 2017, Vienna, Austria, September 7-8, 2017, Revised Selected Papers*, volume 10718 of *Lecture Notes in Computer Science*, pages 127–140. Springer, 2017. doi:10.1007/978-3-319-72751-6_10.

8 Zahra Derakhshandeh, Shlomi Dolev, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. Brief announcement: amoebot - a new model for programmable matter. In Guy E. Blelloch and Peter Sanders, editors, *26th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA '14, Prague, Czech Republic - June 23 - 25, 2014*, pages 220–222. ACM, 2014. doi:10.1145/2612669.2612712.

9 Zahra Derakhshandeh, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. An algorithmic framework for shape formation problems in self-organizing particle systems. In Faramarz Fekri, Sasitharan Balasubramaniam, Tommaso Melodia, Ahmad Beirami, and Albert Cabellos, editors, *Proceedings of the Second Annual International Conference on Nanoscale Computing and Communication, NANOCOM' 15, Boston, MA, USA, September 21-22, 2015*, pages 21:1–21:2. ACM, 2015. doi:10.1145/2800795.2800829.

10 Zahra Derakhshandeh, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. Universal shape formation for programmable matter. In Christian Scheideler and Seth Gilbert, editors, *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2016, Asilomar State Beach/Pacific Grove, CA, USA, July 11-13, 2016*, pages 289–299. ACM, 2016. doi:10.1145/2935764.2935784.

11 Zahra Derakhshandeh, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. Universal coating for programmable matter. *Theor. Comput. Sci.*, 671:56–68, 2017. doi:10.1016/j.tcs.2016.02.039.

12 Zahra Derakhshandeh, Robert Gmyr, Thim Strothmann, Rida A. Bazzi, Andréa W. Richa, and Christian Scheideler. Leader election and shape formation with self-organizing programmable matter. In Andrew Phillips and Peng Yin, editors, *DNA Computing and Molecular Programming - 21st International Conference, DNA 21, Boston and Cambridge, MA, USA, August 17-21, 2015. Proceedings*, volume 9211 of *Lecture Notes in Computer Science*, pages 117–132. Springer, 2015. doi:10.1007/978-3-319-21999-8_8.

13 Yuval Emek, Shay Kutten, Ron Lavi, and William K. Moses Jr. Deterministic leader election in programmable matter. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPIcs*, pages 140:1–140:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ICALP.2019.140.

14 Nicolas Gastineau, Wahabou Abdou, Nader Mbarek, and Olivier Togni. Distributed leader election and computation of local identifiers for programmable matter. In Seth Gilbert, Danny Hughes, and Bhaskar Krishnamachari, editors, *Algorithms for Sensor Systems - 14th International Symposium on Algorithms and Experiments for Wireless Sensor Networks, ALGOSENSORS 2018, Helsinki, Finland, August 23-24, 2018, Revised Selected Papers*, volume 11410 of *Lecture Notes in Computer Science*, pages 159–179. Springer, 2018. doi:10.1007/978-3-030-14094-6_11.

15 Hungwen Li and Massimo Maresca. Polymorphic-torus network. *IEEE Trans. Computers*, 38(9):1345–1351, 1989. doi:10.1109/12.29479.

16 Hungwen Li and Quentin F. Stout. Reconfigurable simd massively parallel computers. *Proceedings of the IEEE*, 79(4):429–443, 1991.

17 Giuseppe Antonio Di Luna, Paola Flocchini, Nicola Santoro, Giovanni Viglietta, and Yukiko Yamauchi. Shape formation by programmable particles. In James Aspnes, Alysson Bessani, Pascal Felber, and João Leitão, editors, *21st International Conference on Principles of Distributed Systems, OPODIS 2017, Lisbon, Portugal, December 18-20, 2017*, volume 95 of *LIPIcs*, pages 31:1–31:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.OPODIS.2017.31.

18 Russ Miller, Viktor K. Prasanna-Kumar, Dionisios Reisis, and Quentin F. Stout. Meshes with reconfigurable buses. In *Proceedings of the fifth MIT conference on Advanced research in VLSI*, pages 163–178, 1988.

19 J. Michael Moshell and Jerome Rothstein. Bus automata and immediate languages. *Inf. Control.*, 40(1):88–121, 1979. doi:10.1016/S0019-9958(79)90361-9.

20 Helmut Prodinger. How to select a loser. *Discret. Math.*, 120(1-3):149–159, 1993. doi:10.1016/0012-365X(93)90572-B.

21 Paul W. K. Rothemund and Erik Winfree. The program-size complexity of self-assembled squares (extended abstract). In F. Frances Yao and Eugene M. Luks, editors, *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 459–468. ACM, 2000. doi:10.1145/335305.335358.

22 Jerome Rothstein. Bus automata, brains, and mental models. *IEEE Trans. Syst. Man Cybern.*, 18(4):522–531, 1988. doi:10.1109/21.17370.

23 Tommaso Toffoli and Norman Margolus. Programmable matter: Concepts and realization. *Int. J. High Speed Comput.*, 5(2):155–170, 1993. doi:10.1142/S0129053393000086.

24 Damien Woods, Ho-Lin Chen, Scott Goodfriend, Nadine Dabby, Erik Winfree, and Peng Yin. Active self-assembly of algorithmic shapes and patterns in polylogarithmic time. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, ITCS '13, page 353–354, New York, NY, USA, 2013. Association for Computing Machinery. doi:10.1145/2422436.2422476.

Coloring Drawings Of Graphs*

Christoph Hertrich[1], Felix Schröder[1], and Raphael Steiner[1]

1 **Institute of Mathematics, Technische Universität Berlin**
{hertrich,fschroed,steiner}@math.tu-berlin.de

Abstract

We consider face-colorings of drawings of graphs in the plane. Given a multi-graph G together with a drawing $\Gamma(G)$ in the plane with only finitely many crossings, we define a *face-k-coloring* of $\Gamma(G)$ to be a coloring of the maximal connected regions of the drawing, the *faces*, with k colors such that adjacent faces have different colors. By the 4-color theorem, every drawing of a bridgeless graph has a face-4-coloring. A drawing of a graph is facially 2-colorable if and only if the underlying graph is Eulerian. We show that every graph without degree 1 vertices admits a 3-colorable drawing. This leads to the natural question which graphs G have the property that each of its drawings has a 3-coloring. We say that such a graph G is *facially* 3*-colorable*. We derive several sufficient and necessary conditions for this property: we show that every 4-edge-connected graph and every graph admitting a *nowhere-zero* 3*-flow* is facially 3-colorable. We also discuss circumstances under which facial 3-colorability guarantees the existence of a nowhere-zero 3-flow. On the negative side, we present an infinite family of facially 3-colorable graphs without a nowhere-zero 3-flow. On the positive side, we formulate a conjecture which has a surprising relation to a famous open problem by Tutte known as the 3*-flow-conjecture*. We prove our conjecture for subcubic and for $K_{3,3}$-minor-free graphs.

Related Version A full version of the paper is available at `https://arxiv.org/abs/2008.09692`

1 Introduction

Graph coloring is one of the earliest and most influential branches of graph theory, whose first occurences date back more than 150 years. Maybe the most celebrated problem in graph theory is the 4-color-problem, asking whether the bounded regions of every planar map can be colored using 4 colors such that regions sharing a common border receive different colors. This problem was finally resolved in the positive in 1972 when Appel and Haken [1, 2] presented a computer-assisted proof of the famous 4*-Color-Theorem*, which formally states that the chromatic number of every planar graph is at most four.

In this paper we combine the topics of graph coloring and graph drawing by studying colorings of planar maps arising from drawings of possibly *non-planar* graphs. Formally, a *face-k-coloring* of a drawing Γ is a proper coloring of the dual graph $G^\top(\Gamma)$ of Γ, i.e., a coloring $c\colon \mathcal{F}(\Gamma) \to \{0, \dots, k-1\}$ of the faces such that for any two faces f_1, f_2 which are *adjacent* in Γ (i.e., they share a common segment of an edge), we have $c(f_1) \neq c(f_2)$ (see Figure 1 for an example). Note that in a drawing Γ it might occur that a face f is adjacent to itself, in which case no face-coloring can exist, compare Figure 2.

However, in this case the edge involved in the self-touching must be a bridge of the underlying abstract graph, hence, self-touchings do not occur in drawings of bridgeless graphs. This justifies that most of our results are formulated only for the setting of bridgeless graphs. Using the Four-Color-Theorem, we directly see that four colors are sufficient to face-color bridgeless graphs.

* All three authors were supported by DFG-GRK 2434 Facets of Complexity.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

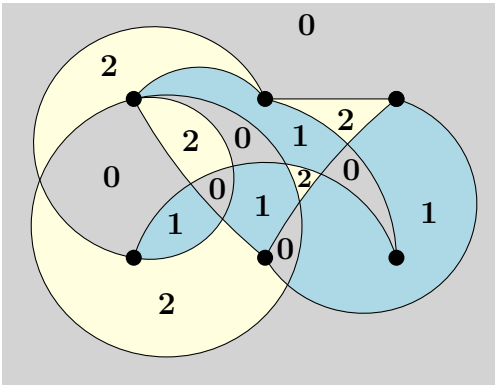


Figure 1 A face-3-colored drawing of a graph.

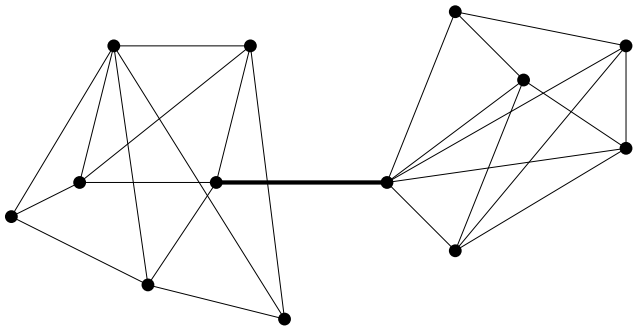


Figure 2 A drawing of a graph with a self-touching outer face, caused by the existence of a bridge (edge marked fat) in the underlying graph.

▶ **Proposition 1.1.** *Every drawing of a bridgeless graph admits a face-4-coloring.*

2 Notation and Basics

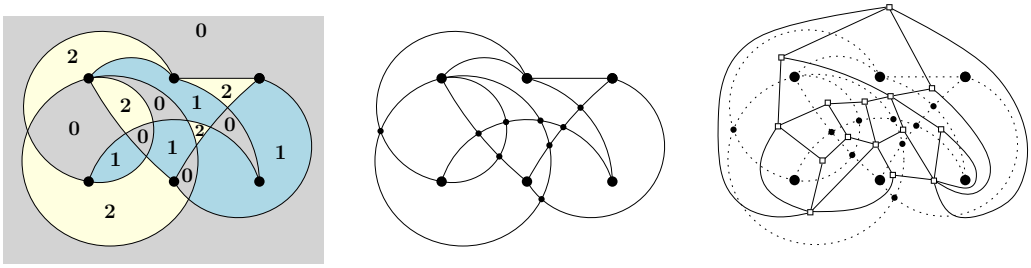
The graphs considered in this paper are finite multi-graphs. We say that G' is a *minor* of G, if G' is obtained from G via a sequence of finitely many edge contractions, edge deletions and vertex deletions. We say that G' is a *subcontraction* of G if it is obtained from G by a sequence of vertex contractions.

By a *drawing* Γ of a graph G we mean an immersion of G in the plane such that vertices are mapped to distinct points and edges are represented by continuous curves connecting the images of their endpoints (closed curves in the case of loops), but which do not contain any other vertices. Edges may intersect each other and themselves, but there are only finitely many points of intersection. For $e \in E(G)$, $\gamma(e)$ indicates the set of points on the curve representing e, and $\gamma^\circ(e)$ the set of interior points of $\gamma(e)$. A more restricted class of drawings are the *good drawings*, satisfying the following additional properties:

- no point is contained in the interiors of more than two edges,
- edges do not self-intersect (except loops may self-intersect at the endpoint)
- every two adjacent edges intersect only in their common endpoints,
- non-adjacent edges intersect in at most one point, which is a proper crossing, and
- loops do not intersect other edges.

Good drawings of simple graphs have been studied extensively in the literature, mainly because the crossing number of a graph is attained by good drawings, see also [8] for a survey on this topic. More precisely, a well-known fact in crossing number theory is that every drawing of a graph can be reduced by a set of local uncrossing-operations to a good drawing. The proof is standard, but lengthy, so we defer it to the full version. A similar proof (which however only deals with simple graphs) can be found in [8], Lemma 1.3.

In case Γ is a drawing of G, we also say that G is the *underlying graph* of Γ.



(a) Drawing Γ of G colored by c (b) The planarization $G_{\text{isc}}(\Gamma)$ (c) $G^{\top}(\Gamma)$, planar dual to $G_{\text{isc}}(\Gamma)$

Figure 3 Face-3-coloring c of a drawing of a multigraph G and associated graphs G_{isc} and $G^{\top}$.

By $\mathcal{F}(\Gamma)$ we denote the set of the *faces* of Γ, i.e., the connected components of $\mathbb{R}^2 - \Gamma$. Placing an additional vertex at every inner intersection of at least two edges in a drawing Γ and making two such vertices adjacent whenever they appear consecutively on the same edge of the drawing, we obtain an embedded planar graph $G_{\text{isc}}(\Gamma)$, which we refer to as the *planarization* of Γ. See Figure 3(b) for an example. Note that $G_{\text{isc}}(\Gamma)$ captures the most relevant combinatorial properties of the drawing Γ. In particular, the cell decompositions of $\mathbb{R}^2 - \Gamma$ and $\mathbb{R}^2 - G_{\text{isc}}(\Gamma)$ are isomorphic. By $G^{\top}(\Gamma)$ we will denote the planar dual graph of $G_{\text{isc}}(\Gamma)$, embedded in the natural way, see Figure 3(c) for an example.

Wherever the proof of any result is not given in this extended abstract or not given to its full extent, it can be found in the full version.

3 Existence of 2- and 3-Colorable Drawings

We start our investigation of face-colorings of graphs by characterising the drawings whose faces can be properly colored using only two colors. A graph is called *Eulerian* if all its vertices have even degree.

▶ **Proposition 3.1.** *A drawing Γ of a graph G has a face-2-coloring if and only if G is Eulerian.*

Proof Sketch. If G is Eulerian, the planarization of Γ is as well, and the dual of a planar, Eulerian graph is bipartite. Therefore, Γ has a face-2-coloring. On the other hand, the regions around any vertex of odd degree form an odd cycle which cannot be 2-colored. ◀

For every face-coloring of a drawing of a non-Eulerian graph, at least 3 colors are required. However, we show that this is the worst case: every graph without degree one vertices (in particular, any bridgeless graph) has numerous drawings that are 3-colorable.

▶ **Proposition 3.2.** *A graph G has a drawing with a face-3-coloring if and only if it has no vertex of degree 1.*

Propositions 1.1, 3.1, and 3.2 motivate the problem of understanding the structure of graphs all whose drawings are 3-colorable. This leads to the following notion: If G is an abstract graph, we say that G is *facially* 3*-colorable* if every drawing of G in the plane admits a face-3-coloring.

4 Sufficient Conditions for Facial 3-Colorability

We derive several sufficient conditions for a graph to be facially 3-colorable, and thereby draw a link between facially 3-colorable graphs and so-called 3-*flowable* graphs, which are intensively studied in the theory of *nowhere-zero flows* on graphs. For $k \in \mathbb{N}$, a *nowhere-zero* k*-flow* on a graph G consists of a pair (D, f), where $D = (V(D), A(D))$ is an orientation of the edges of G, and where $f\colon A(D) \to \mathbb{Z}_k \setminus \{0\}$ is a group-valued flow on the digraph D, i.e., a weighting of the arcs with non-zero group elements from $\mathbb{Z}_k$ satisfying Kirchhoff's law of flow conservation:

$$\forall v \in V(D)\colon \sum_{e=(w,v)\in A(D)} f(e) = \sum_{e=(v,w)\in A(D)} f(e)$$

If a graph G admits a nowhere-zero k-flow, we also say that G is k*-flowable*. The interest in nowhere-zero-flows stems from the following intimate connection to colorings of planar graphs. For a comprehensive introduction to the topic of nowhere-zero flows, we refer to the textbook [13] by Zhang. Particularly relevant to the topics addressed here are the sections 'Face Colorings' and 'Nowhere-Zero 3-Flows'.

▶ **Theorem 4.1** (Folklore, see also [13], Theorem 1.4.5)**.** *Let G be a planar graph and let Γ be a crossing-free embedding of G in the plane. Then for any $k \in \mathbb{N}$, G admits a nowhere-zero k-flow if and only if Γ has a face-k-coloring.*

Similar to the situation for face-colorings, only bridgeless graphs can have nowhere-zero flows, as the flow value of a bridging edge must be 0. Conversely, a famous result by Seymour [9] states that every bridgeless graph admits a nowhere-zero 6-flow. The following result relates the existence of nowhere-zero 3-flows in graphs with the existence of face-3-colorings for all their drawings.

▶ **Theorem 4.2.** *Let G be a graph admitting a nowhere-zero 3-flow. Then G is facially 3-colorable.*

Grötzsch's theorem states that loopless triangle-free planar graphs are 3-colorable, thus we obtain another interesting positive result.

▶ **Theorem 4.3.** *Every 4-edge-connected graph is facially 3-colorable.*

Proof idea. Let G be a 4-edge-connected graph. We first prove that the planarization of any drawing of G is 4-edge-connected as well. Note that this means that its planar dual graph is simple and triangle-free. Hence, Grötzsch's Theorem implies that the dual graph admits a proper 3-vertex-coloring, and therefore the faces can be properly colored with 3 colors. ◀

5 An Infinite Family of Counterexamples

Looking at Theorem 4.2 and 4.3, it is natural to ask whether there are facially 3-colorable graphs that are not 3-flowable. We answer this question in the positive by providing an

infinite family of graphs with this property. For every $n \in \mathbb{N}$, $K^+_{3,n}$ denotes the graph obtained from the complete bipartite graph $K_{3,n}$ by joining two vertices in the partite set of size 3 by an edge.

▶ **Theorem 5.1.** *For every $n \geq 4$, the graph $K^+_{3,n}$ is facially 3-colorable but does not admit a nowhere-zero 3-flow.*

However, examples of graphs as given by Theorem 5.1 are rarely spread. In fact, Sudakov [10] proved that random graphs expected to have minimum degree at least 2 are expected to have a nowhere-zero 3-flow, thereby establishing in a strong sense that almost all graphs admit a nowhere-zero 3-flow.

6 Towards Characterizing Facially 3-Colorable Graphs

The following two results show that an equivalence between facial 3-colorability and the existence of nowhere-zero 3-flows holds at least for sparse graph classes beyond planar graphs. The proofs rely on the fact that facial 3-colorability is hereditary with respect to subcontractions, that is, contractions of arbitrary subsets of vertices (not only connected subgraphs).

▶ **Theorem 6.1.** *A graph with maximum degree at most 3 is facially 3-colorable if and only if it is 3-flowable.*

▶ **Theorem 6.2.** *A $K_{3,3}$-minor-free graph is facially 3-colorable if and only if it is 3-flowable.*

We have not been able to find graphs which are facially 3-colorable but not 3-flowable except for graphs arising by simple operations from the examples given by Theorem 5.1. To be more precise, we believe that excluding the graphs $K^+_{3,n}$, $n \geq 4$, as subcontractions could already be sufficient to yield an equivalence between facial 3-colorability and 3-flowability.

▶ **Conjecture 6.3.** *If G is a facially 3-colorable graph which does not have a subcontraction isomorphic to $K^+_{3,n}$ for some $n \geq 4$, then G is 3-flowable.*

Interestingly, a positive answer to Conjecture 6.3 would also imply a positive answer to the following long-standing Conjecture by Tutte.

▶ **Conjecture 6.4** (Tutte's 3-Flow-Conjecture, Conjecture 1.1.8 in [13])**.** *Every 4-edge-connected graph admits a nowhere-zero 3-flow.*

To see this, suppose Conjecture 6.3 holds true, and let G be a given 4-edge-connected graph. By Theorem 4.3, G is facially 3-colorable. Since G is 4-edge-connected, so is each of its subcontractions. Since each $K^+_{3,n}$ has a vertex of degree 3, this means that G has no subcontraction isomorphic to a $K^+_{3,n}$ with $n \geq 4$. Hence, Conjecture 6.3 yields that G is 3-flowable, as claimed in Tutte's conjecture.

7 Conclusive Remarks

Apart from the obvious challenges to decide Conjecture 6.3 and to obtain a better understanding of the class of facially 3-colorable graphs, we have an interesting open question towards the computational complexity of recognizing facially 3-colorable graphs. For planar graphs, NP-completeness can be deduced as follows.

▶ **Corollary 7.1.** *Deciding whether a given planar graph is facially* 3*-colorable is* NP*-complete.*

Proof. Testing whether a planar graph is facially 3-colorable by Theorem 6.2 is equivalent to testing whether a given planar graph is 3-flowable. This problem is clearly contained in the class NP (we can verify a nowhere-zero 3-flow in polynomial time). By Theorem 4.1 and planar duality, we can furthermore reduce the problem of deciding whether a given planar graph is properly 3-vertex-colorable to this problem. Since this problem is NP-complete (see [4]), we deduce the claim. ◀

While this clearly suggests hardness for general graphs as well, containment in NP remains unclear, since facial 3-colorability cannot be verified via 3-flowability in this case.

▶ **Question 7.2.** *Is deciding whether an input graph is facially* 3*-colorable contained in* NP*?*

Acknowledgements We are indebted to Andrew Newman, Günter Rote, and László Kozma, who have been involved in fruitful discussions during the Problem Solving Workshop of the Research Training Group 'Facets of Complexity' at Kloster Chorin in December 2019.

References

1 K. Appel and W. Haken. Every Planar Map is Four Colorable. I. Discharging. *Illinois Journal of Mathematics*, 21(3):429–490, 1976.
2 K. Appel, W. Haken, and J.Koch. Every Planar Map is Four Colorable. II. Reducibility. *Illinois Journal of Mathematics*, 21(3):491–567, 1977.
3 C. N. da Silva and C. L. Lucchesi. Flow-Critical Graphs. *Electronic Notes in Discrete Mathematics*, 30:165–170, 2008.
4 M. R. Garey and M. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. H. Freeman and Co., 1990. doi:10.5555/574848.
5 H. Grötzsch. Zur Theorie der diskreten Gebilde, VII: Ein Dreifarbensatz für dreikreisfreie Netze auf der Kugel. *Wiss. Z. Martin-Luther-U., Halle-Wittenberg, Math.-Nat. Reihe*, 8:109–120, 1959.
6 C. Hertrich, F. Schröder, and R. Steiner. Coloring drawings Of Graphs, 2020. arXiv:2008.09692.
7 J. Li, Y. Ma, Y. Shi, W. Wang, and Y. Wu. 3-flow critical graphs, 2020. arXiv:2003.09162.
8 M. Schäfer. *Crossing numbers of graphs*. Discrete Mathematics and Its Applications. CRC Press, 2017.
9 P. D. Seymour. Nowhere-zero 6-flows. *Journal of Combinatorial Theory, Series B*, 30(2):130–135, 1981.
10 B. Sudakov. Nowhere-Zero Flows in Random Graphs. *Journal of Combinatorial Theory, Series B*, 81(2):209–223, 2001.
11 K. Wagner. Über eine Eigenschaft der ebenen Komplexe. *Mathematische Annalen*, 114:570–590, 1937.
12 D. J. A. Welsh. Euler and bipartite matroids. *Journal of Combinatorial Theory*, 6(4):375–377, 1969.
13 C.-Q. Zhang. *Integer Flows and Cycle Covers of Graphs*. Pure and Applied Mathematics. CRC Press, 1997.

Local Complexity of Polygons

Fabian Klute[*1], **Meghana M. Reddy**[†‡2], **and Tillmann Miltzow**[§1]

1 **Utrecht University, Information and Computing Science Department**
`f.m.klute@uu.nl`, `t.miltzow@googlemail.com`
2 **ETH Zürich, Department of Computer Science**
`meghana.mreddy@inf.ethz.ch`

Abstract

Many problems in Discrete and Computational Geometry deal with simple polygons or polygonal regions. Many algorithms and data-structures perform considerably faster if the underlying polygonal region has low local complexity. One obstacle to make this intuition rigorous, is the lack of a formal definition of local complexity. Here, we give two possible definitions and show how they are related in a combinatorial sense. We say that a polygon P has *point visibility width* $\texttt{pvw}(P)$ (or `pvw` in short), if there is no point in P that sees more than `pvw` reflex vertices. We say that a polygon P has *chord visibility width* $\texttt{cvw}(P)$ (or `cvw` in short) if there is no chord (maximal segment) in P that sees more than `cvw` reflex vertices. We show that

$$\texttt{cvw} \leq \texttt{pvw}^{O(\texttt{pvw})},$$

for any simple polygon. Furthermore, we show that there exists a family of simple polygons with

$$\texttt{cvw} \geq 2^{\Omega(\texttt{pvw})}.$$

Related Version `arxiv.org/abs/2101.07554`

1 Introduction

In Discrete and Computational Geometry we study many problems with respect to the input size n and other natural parameters that capture some properties of the problem. One famous example is the computation of the convex hull of a set of points in the plane. While $\Theta(n \log n)$ time is worst case possible, this can be improved to $\Theta(n \log h)$, where h is the number of vertices on the convex hull [4]. Here, the number of vertices on the convex hull is a natural parameter to study this problem. We also say sometimes that the algorithm is output-sensitive. Another famous example, is the spread Δ of a set of points in the plane. That is the ratio between the largest and the smallest distance, between any two points. Efrat and Har-Peled were the first to find an approximation algorithm for the art gallery problem under the assumption that the underlying set of vertices has bounded spread [2]. A third example is the number of reflex vertices of a polygon. This parameter gave raise to some FPT algorithms for the art gallery problem [1].

In this work, we introduce two new parameters that are meant to capture rigorously the idea of local complexity. Consider the polygons shown in Figure 1. Most researchers would probably agree that the polygon on the left has lower local complexity than the polygon on the right. Yet, it is not straightforward how to define this rigorously in a mathematical sense.

* Supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 612.001.651.
† Supported by the Swiss National Science Foundation within the collaborative DACH project *Arrangements and Drawings* as SNSF Project 200021E-171681.
‡ The second author's full last name consists of two words and is *Mallik Reddy*. However, she consistently refers to herself with the first word of her last name being abbreviated.
§ Supported by the NWO Veni project EAGER.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

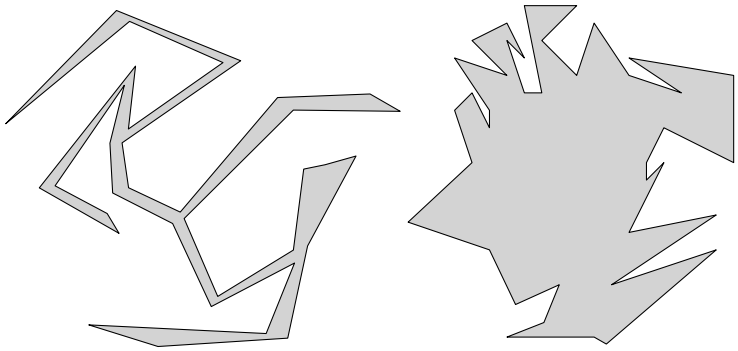


Figure 1 The polygon on the left has intuitively lower local complexity than on the right. Note that with respect to other measures the left figure might appear more complex.

Here, we give two possible definitions and show how they are related in a combinatorial sense. We say that a polygon P has *point visibility width* $\texttt{pvw}(P)$ (or pvw if P is clear from the context), if pvw is the smallest number such that every point in P sees at most pvw reflex vertices. We say that a polygon P has *chord visibility width* $\texttt{cvw}(P)$ (or cvw if P is clear from the context), if cvw is the smallest number such that every chord (maximal segment) in P sees at most cvw reflex vertices.

We show the following theorem.

▶ **Theorem 1.1.** *For every polygon with chord visibility width* cvw *and point visibility width* pvw*, it holds that*

$$\texttt{pvw} \leq \texttt{cvw} \leq \texttt{pvw}^{O(\texttt{pvw})}.$$

Moreover, there are polygons such that

$$\texttt{cvw} \geq 2^{\Omega(\texttt{pvw})}.$$

Note that Hengeveld and Miltzow already defined the notion of chord visibility width in a very similar way [3]. Their definition counts the total number of seen vertices instead of the total number of reflex vertices. They showed that the art gallery problem admits an FPT algorithm with respect to chord visibility width. Note that their FPT algorithm also works verbatim also with our definition. Now, Theorem 1.1 implies that there is also an FPT algorithm for the art gallery problem with respect to the point visibility width. Note however, that the running time becomes double exponential.

For a parameter to be interesting to study, we usually have three criteria.

naturalness: Although there is no definition of what it means to be mathematically natural, many researchers seem to have a common understanding of this notion.
relevance: The parameter is at least for some fraction of instances reasonably low.
profitable: Using the parameter, we should be able to design better algorithms and prove useful run time upper bounds.

We believe that both parameters are mathematically natural. Theorem 1.1 indicates that the chord visibility width can be exponentially larger than the point visibility width. Thus we would expect that chord visibility width is potentially more profitable. The aforementioned FPT results indicate that chord visibility width is indeed profitable. Having a double exponential time FPT algorithm can barely be called a useful algorithm. We would expect that both parameters are equally relevant as the example that we give is fairly contrived. The remainder of this paper is dedicated to proving Theorem 1.1.

2 Chord visibility width vs Point visibility width

We prove Theorem 1.1 in two parts. First, we show the second half of the theorem in Section 2.1 by constructing a polygon for which it holds that $\texttt{cvw} \geq 2^{\Omega(\texttt{pvw})}$. Second, we show the first half of Theorem 1.1 in Section 2.2 by analysing how the reflex vertices visible from a chord in a simple polygon P restrict each others vision and relating this to the point visibility width of the polygon.

2.1 Lower bound

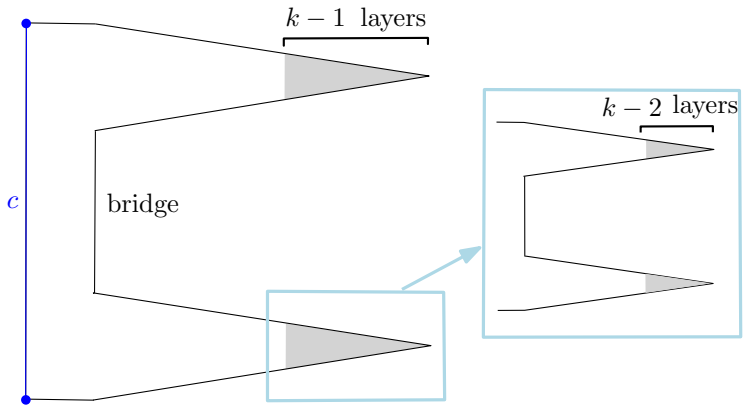


Figure 2 Construction of the Iterated Comb.

We construct a polygon P, called the *Iterated Comb*, see Figure 2. In the following, let $k \in \mathbb{N}$. The Iterated Comb consists of k *layers*, each layer consists of two *spikes* and each spike further splits into two more spikes in the subsequent layer. Observe, that the entire polygon is visible from the chord connecting the two left-most points of the polygon. The distance between consecutive spikes in a layer, referred to as the *bridge*, is adjusted such that if at least one vertex in the interior of a spike is visible from a point p on c, then p cannot see any interior vertex of any other spike. This property is achieved by stretching the bridges vertically. More specifically, for $1 < i \leq k$, the length of the bridge of the i^{th} layer is increased such that the property holds for layer i and then the bridge of the previous layer is adjusted accordingly. By iteratively stretching the bridges from the last layer to the first layer, it can be ensured that the property holds for every layer. This property is illustrated in Figure 3 for $k = 2$. In the first layer, the point p sees an interior vertex of the first spike and no interior vertex of the second spike. Similarly, in the second layer point p sees an interior vertex of the second spike and no interior vertex of the first spike.

Chord visibility width of the Iterated Comb

Clearly, the chord which sees the highest number of reflex vertices is the chord defined by the two left-most vertices. Let this chord be c. The number of reflex vertices of the first layer visible from c is two. Similarly, the number of reflex vertices of the i^{th} layer visible from c is 2^i. Summing up over all k layers, the number of reflex vertices visible from c is $\Theta(2^{k+1})$, and hence $\texttt{cvw} = \Theta(2^{k+1})$.

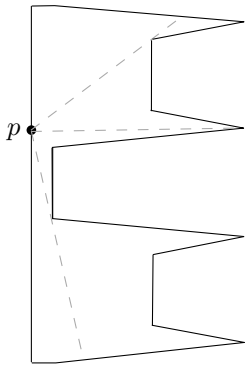


Figure 3 Point p sees interior points of at most one spike of any layer

Point visibility width of the Iterated Comb

▶ Claim 1. Chord c contains at least one of the points in P which see the highest number of reflex vertices of P.

Proof. Let q be a point in polygon P which sees the highest number of reflex vertices of P. Let p be a point on chord c which has the same y-coordinate as q. Assume $p \neq q$. Let r be a reflex vertex visible from q. Since P is monotone with respect to y-axis, the triangle pqr must be empty. This implies that r is visible from p as well. Hence, the point p also sees the highest number of reflex vertices in P since p sees at least as much as q. Refer to Figure 4 for an illustration. ◀

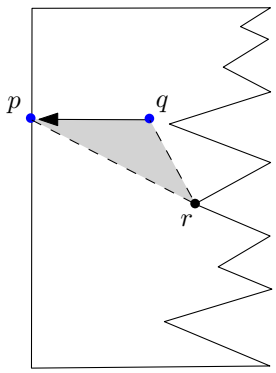


Figure 4 Point p sees all the reflex vertices visible from q

Let p be any point on c. Both the reflex vertices in layer one are visible from p. In each subsequent layer, p can see the reflex vertices that are in the interior of at most one spike, which is two reflex vertices, p cannot see any of the other reflex vertices in the other spikes by construction. Summing it up, we can conclude that $2k$ reflex vertices are visible from p, and thus $\texttt{pvw} = 2k$. Hence the Iterated Comb has $\texttt{cvw} \geq 2^{\Omega(\texttt{pvw})}$.

2.2 Upper bound

Next, we show that we can upper bound the chord visibility width in terms of the point visibility width.

To this end, we prove the following lemma.

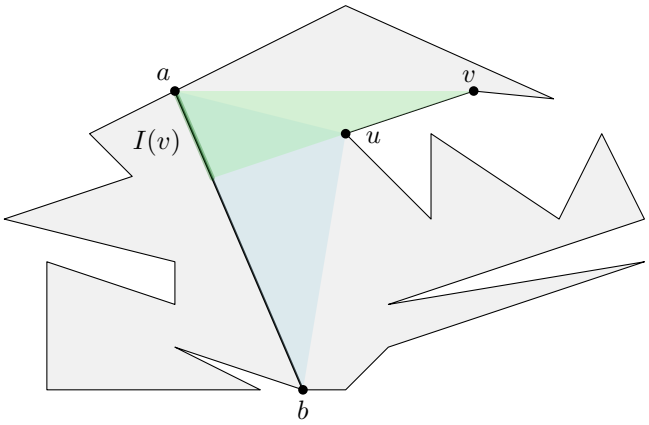


Figure 5 The vertex v sees a subinterval $I(v) \subseteq s$ which is restricted by a and u.

▶ **Lemma 2.1.** *For every simple polygon, it holds that*

$$\texttt{cvw} \leq \texttt{pvw}^{O(\texttt{pvw})}.$$

The rest of this paragraph is dedicated to the proof of Lemma 2.1.

For that purpose assume, we are given a simple polygon P together with a chord $s \subset P$. Furthermore, we assume that no point in P sees more than $k = \texttt{pvw}$ reflex vertices of P. Let us denote by R the set of all reflex vertices that see at least one point of $s = \text{seg}(a, b)$. Furthermore, we also include the two endpoints of s in the set R. As P is a simple polygon it holds that every reflex vertex $r \in R \setminus \{a, b\}$ sees a subsegment $I(r) \subseteq s$. For convenience, we also call $I(r)$ an *interval*.

Note that every interval is *restricted* by exactly two points in R, see Figure 5. In case of ambiguity, due to collinearities, we say the point in R closer to s is the restricting point. Those vertices can be either the endpoints of s (a and b) or a different reflex vertex in R. We show the following claim.

▶ Claim 2. If u is a reflex vertex that restricts the reflex vertex v then it holds that

$$I(v) \subseteq I(u).$$

Proof. The triangle T formed by $I(v)$ and v is trivially convex and fully contained inside P. The reflex vertex u is on the boundary of the triangle and thus sees every point of T. In particular also $I(v)$. ◀

Given the previous claim, we construct the visibility restriction graph G as follows. The vertices are formed by the points in R. We say that uv forms a directed edge, if u is restricted by v. We summarize a few useful properties of G in the following claim.

▶ Claim 3. The visibility restriction graph of a polygon with point visibility width at most k has the following properties.

1. The segment endpoints a, b are the only two sinks.
2. The out-degree is two for every vertex $v \in R \setminus \{a, b\}$.
3. The in-degree is at most $k - 1$ for every vertex $v \in R$.
4. The longest path has at most $k + 1$ vertices.

Proof. By definition, every reflex vertex is restricted by exactly two vertices in R. This implies Item 1 and 2.

Any reflex vertex v can see itself and all its neighbors. Its in-degree neighbors are also reflex vertices. As no point can see more than k reflex vertices v has at most $k-1$ in-degree neighbors. This concludes the proof of Item 3.

Finally, to prove Item 4, let $p = u_1 u_2 \dots u_l$ be a directed path. Then it holds that there is a point

$$q \in I(u_l) \subseteq \dots \subseteq I(u_2) \subseteq I(u_1) = s.$$

The point q sees all reflex vertices of the path p. As no point sees more than k reflex vertices, it holds that p has at most k reflex vertices. As all but potentially the first vertex is a reflex vertex, we have $l \leq k+1$. ◀

The properties of the last claim enable us to give an upper bound on the size of G and thus also on the size of R.

▶ Claim 4. The visibility restriction graph G of a polygon with point visibility width $\texttt{pvw} = k$ has at most $k^{O(k)}$ vertices.

Proof. We organize G into layers depending on the distance from a and b. Note that if layer i has t vertices then layer $(i+1)$ has at most $t \cdot k$ vertices. As there are at most $k+1$ layers and the first layer has size two we get that G has at most

$$2 + 2k + 2k^2 + 2k^3 + \dots + 2k^k = k^{O(k)}$$

vertices. ◀

3 Conclusion

We believe that local complexity has the potential to be a useful parameter. We gave two ways to define local complexity in a rigorous way and showed how those two ways relate to one another. We want to end with a few open questions.

1. Can we find algorithms and data structures that can make use of low local complexity?
2. Can we compute or approximate the point visibility width and chord visibility width in an efficient manner? Note that this is more a theoretical question. We do not necessarily need to know the chord visibility width of a polygon to use the concept in the design and analysis of an algorithm.
3. Are there other ways to formalize the idea of local complexity within a polygonal region?

References

1 Akanksha Agrawal and Meirav Zehavi. Parameterized analysis of art gallery and terrain guarding. In *International Computer Science Symposium in Russia*, volume 12159 of *LNCS*, pages 16–29. Springer, 2020. doi:10.1007/978-3-030-50026-9_2.

2 Alon Efrat and Sariel Har-Peled. Guarding galleries and terrains. *Information Processing Letters*, 100(6):238–245, 2006. doi:https://doi.org/10.1016/j.ipl.2006.05.014.

3 Simon Hengeveld and Tillmann Miltzow. A practical algorithm with performance guarantees for the art gallery problem. In *SoCG*, 2021. https://arxiv.org/abs/2007.06920.

4 David G. Kirkpatrick and Raimund Seidel. The ultimate planar convex hull algorithm? *SIAM Journal on Computing*, 15(1):287–299, 1986. doi:10.1137/0215021.

Minimum Link Fencing*

Sujoy Bhore[1], Fabian Klute[2], Maarten Löffler[2], Soeren Nickel[3], Martin Nöllenburg[3], and Anaïs Villedieu[3]

1 **Université Libre de Bruxelles**
`sujoy.bhore@gmail.com`
2 **Utrecht University**
`[f.m.klute|m.loffler]@uu.nl`
3 **TU Wien**
`[soeren.nickel|noellenburg|avilledieu]@ac.tuwien.ac.at`

Abstract

We study a variant of the geometric multicut problem, where we are given n colored and pairwise interior-disjoint polygons $\mathcal{P}$ in the plane. The objective is to compute a set of simple closed polygon boundaries (fences) that separate the polygons in such a way that any two polygons which are enclosed by the same fence have the same color, and the total number of links in the fencing is minimized. We call this the *minimum link fencing* (MLF) problem. In this work we primarily consider a special case of MLF, called *bounded minimum link fencing* (BMLF), where $\mathcal{P}$ contains a polygon Q that is not bounded in any direction, and behaves as an outer polygon. We show that BMLF is NP-hard in general, however, it is polynomial-time solvable when the convex hull of the input polygons except Q does not intersect Q.

1 Introduction

In the *geometric multicut* problem [2], one is given k disjoint sets in the plane, each with a different *color*, and asks for a subdivision of the plane such that no cell of the subdivision contains multiple colors. The goal is to minimize the total length of the subdivision edges.

* FK was supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 612.001.651; SB, MN, and AV were supported by the Austrian Science Fund (FWF) under grant P31119.

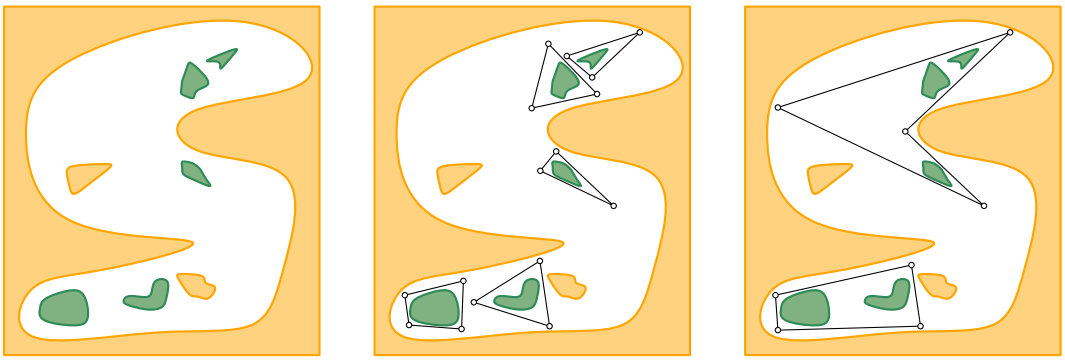
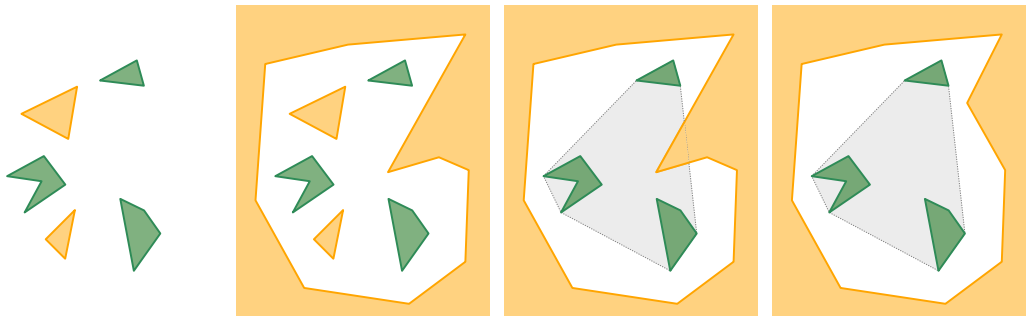


Figure 1 Two sets of polygons in the plane (left) with different colors (green and yellow). The yellow set effectively acts as an outer polygon with holes. Separating the two sets with individual fences (middle) can lead to significantly more links in the fences (here 16) than grouping same-colored polygons (right), which manages this in just 8 links.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.



(a) MLF **(b)** BMLF **(c)** SMLF **(d)** CMLF

Figure 2 Different problem inputs corresponding to (a) MLF, (b) BMLF, (c) SMLF and (d) CMLF. In (c) and (d) the convex hull of all inner polygons is indicated in grey.

A different kind of separation is achieved in the *polygon nesting* problem [4], where for two polygons P and Q with $P \subset Q$ one asks for the polygon P' with the smallest number of links, s.t. $P \subset P' \subset Q$. There exists a series of work that adressed the algorithmic complexity of nesting problems for various polygon families; see [4, 5, 7, 9, 10].

We consider a variant of geometric multicut inspired by polygon nesting, where we separate the subsets from each other, with a set of closed polygons (fences), which enclose only polygons of one color and have the smallest possible number of links. If one or more sets are not connected, we need to solve the combinatorial problem of choosing which polygons should be grouped in each fence polygon. Figure 1 illustrates the problem. Some variants of the fencing problem already become NP-hard for point objects with two colors, e.g., if we require the fence to be a single closed curve [6].

In this paper we assume the input sets are collections of polygons, one color covers the plane minus a single polygonal hole (the outer polygon, a parallel to polygon nesting) and we will focus on the case $k = 2$. We use n to denote the total number of corners of the input polygons. Even in this simple setting the problem is non-trivial. If both sets are connected, then the problem is equivalent to finding a minimal *nested* polygon, which can be solved in $O(n \log n)$ time [3]. If both sets are not connected we show this problem to be NP-hard in Section 2. Note the contrast to the geometric multicut problem, which is polynomially solvable for $k = 2$ [1] but becomes NP-hard when $k = 3$ [2]. Finally, in Section 3, we show that the problem is polynomial-time solvable when the convex hull of the second color (the *inner polygons*) is contained in the outer polygon and the first color is connected.

1.1 Definitions and Notation

▶ **Definition 1.1** (Minimum Link Fencing (MLF))**.** We are given n pairwise interior-disjoint polygons $\mathcal{P} = \{P_1, \ldots, P_n\}$ in the plane, with a coloring function $f : \mathcal{P} \to \{1, \ldots, k\}$, which assigns a color to every input polygon. We write $\mathcal{P}_i = \{P \mid f(P) = i\}$. We want to find a set of simple closed polygon boundaries $\mathcal{F} = \{F_1, \ldots, F_m\}$ such that the total number of links $|F|$ on the boundary of $F = \bigcup_{i=1}^{k} F_i$ is minimized and if two polygons P_a and P_b are enclosed by the same fence or are both in $\mathbb{R}^2 \setminus \bigcup_{i=1}^{k} \overline{F_i}$, where $\overline{F_i}$ is the polygon bounded by F_i, then $f(P_a) = f(P_b)$. We will call F_i a fence and $\mathcal{F}$ a *minimum link fencing*.

We refer to $\mathcal{P}$, which contains polygons of k different colors, as k-colored and to the problem setting as the k-colored problem. We consider several problem variations.

If there exists a polygon $Q \in \mathcal{P}$, which is unbounded in every direction, i.e., $\mathbb{R}^2 \setminus Q$ is finite,

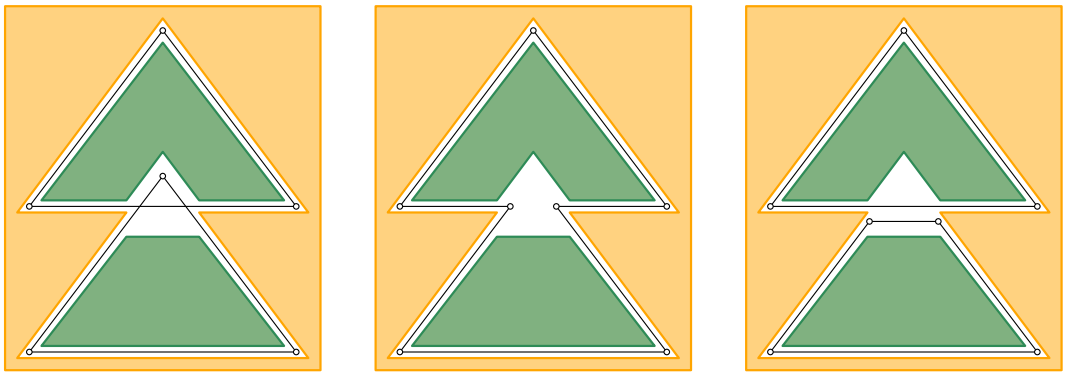


Figure 3 On this input allowing overlapping fences (left) achieves six links, while non-overlapping fences require at least seven links, whether grouping the inner polygons (middle) or not (right).

this subset effectively acts as an outer boundary. In this case we call the problem Bounded Minimum Link Fencing (BMLF). We denote the polygon Q as the *outer polygon*. If Q is the only polygon of its color $f(Q)$ we call this setting Simply Bounded Minimum Link Fencing (SMLF). Moreover, if in an instance of SMLF we have $CH(\bigcup_{i=1}^{k} \mathcal{P}_i \setminus Q) \subset \mathbb{R}^2 \setminus Q$, i.e., the convex hull of all input polygons except Q does not intersect Q, we speak of Convex Bounded Minimum Link Fencing (CMLF). The differences between these settings are illustrated in Fig. 2.

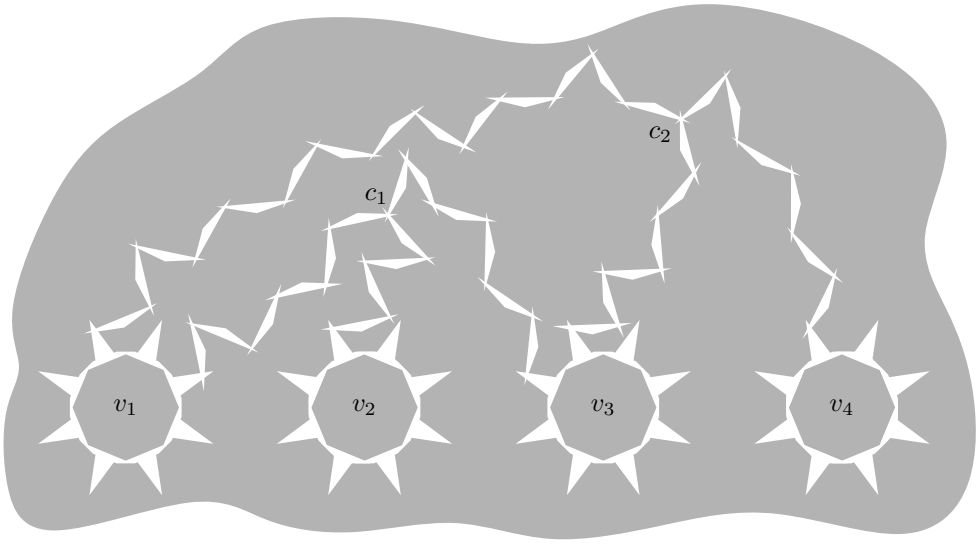
If $\mathcal{F}$ is required to be a set of pairwise crossing-free boundaries, we will prepend *disjoint* to the problem name, i.e, disjoint *MLF rather than just *MLF, where *MLF stands for MLF, BMLF, SMLF, CMLF or any other variant of the problem. Any solution for disjoint *MLF is also a solution for *MLF, however as shown in Fig. 3 a solution for *MLF can achieve fewer links than for disjoint *MLF on the same input. A version of disjoint MLF, which restricts every fence F_i to enclose exactly one polygon $P_i \in \mathcal{P}$, was proven NP-hard by Das in his doctoral thesis [5].

2 Two-colored BMLF is NP-hard

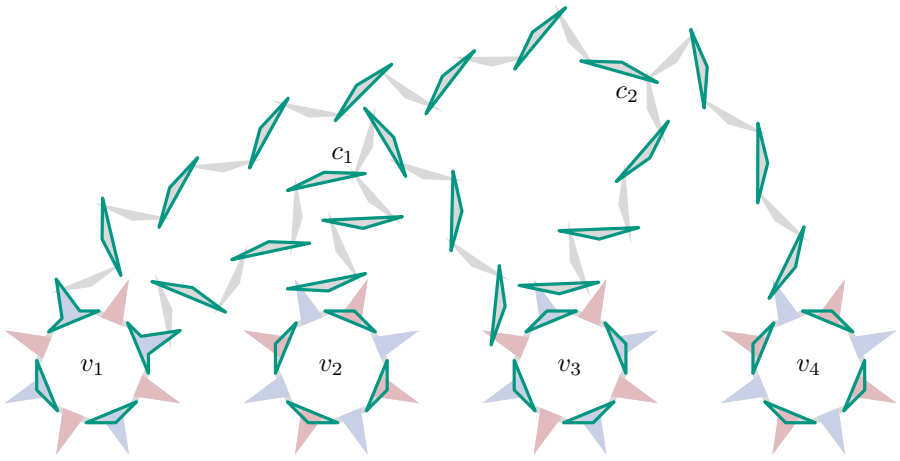
In this section we will call polygons of color one *boundary polygons* and polygons of color two *inner polygons*. An instance of planar 3,4-SAT, which is NP-complete [8], consists of a Boolean CNF-formula ϕ with a set of variables $\mathcal{V} = \{v_1, \ldots, v_n\}$ and a set of clauses $\mathcal{C} \subset 2^{\mathcal{V}}$, s.t. every clause is a disjunction of three literals and every variable occurs at most four times as a literal in a clause. Additionally, we are given the embedded plane incidence graph $G_\phi = (\mathcal{V} \cup \mathcal{C}, E)$. The edge vc is contained in E if v occurs in c as a literal.

We create an instance of 2-colored BMLF, emulating the shape of the embedding of G_ϕ with one unbounded outer polygon Q and multiple boundary polygons of the same color $f(Q) = 1$, s.t., ϕ is satisfiable if and only if the instance has a minimum link fencing with a fixed number of links. This is shown in Fig. 4a, where the unbounded polygon and inner polygons of color 1 are depicted in grey. To ease depiction, we will resort to describing the construction of the free space, rather than the actual polygons. At the end of the reduction all grey polygons need to be subtracted from the plane, to obtain the actual polygons of color 1. Note how the white area in Fig. 4a corresponds to the grey area in Fig. 4b.

▶ **Theorem 2.1.** *Two-colored BMLF is* NP*-hard, even if every fence in $\mathcal{F}$ encloses at most three inner polygons.*

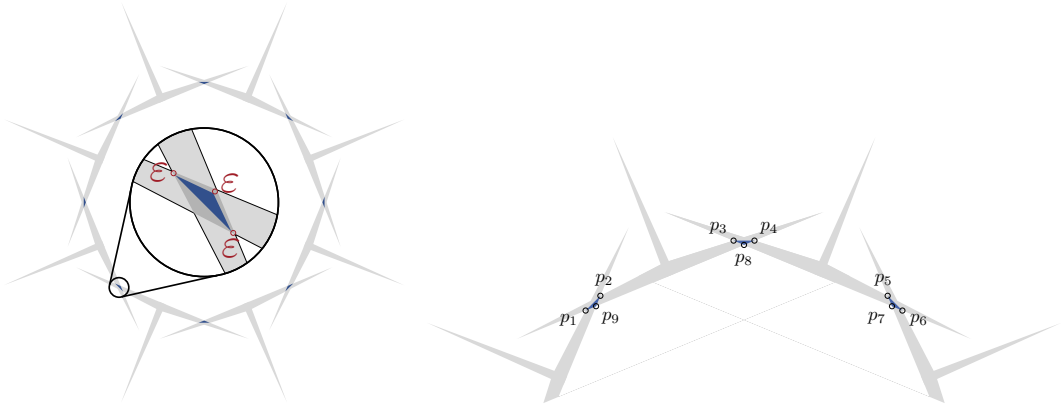


(a) Schematized reduction construction



(b) Schematized construction, with a valid fencing

Figure 4 A (schematized) construction (a) and valid fencing of this construction (b) for a small instance $(v_1 \vee v_3 \vee v_4) \wedge (v_1 \vee v_2 \vee \neg v_3)$. The gray area in (a) indicates the unbounded outer polygon as well as inner polygons of the same color (holes). Inner polygons of the second color are placed in the white area. In (b) fences are highlighted in green and the spikes of the variable gadgets are highlighted as true (blue) of false (red) spikes. Note that the wire $\mathcal{G}(v_3, c_2)$ is in a false state, since c_2 is already fulfilled by v_1 and false state wires save one link over true state wires. The grey area in (b) correspond to the free space of the construction.



(a) Overlapping T-polygons. **(b)** Labeling of chocke points of inner polygons (triangles).

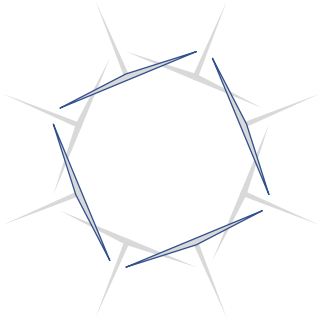
Figure 5 Variable gadget construction. The inner triangles create ε-sized choke points with the outer polygon, highlighted in (a).

Variable gadget. For every variable $v \in \mathcal{V}$, we construct a variable gadget $\mathcal{G}(v)$ as a circular arrangement of eight overlapping T-shaped polygons, see Fig. 5a. The boundary of the union of these polygons defines a boundary polygon in the middle of the variable gadget and the unbounded outer polygon outside of it. Every such T-polygon has an isosceles triangle as the *arm* (the horizontal stroke at the top) of the T and a *spike* protruding from the arm and two consecutive polygons overlap at the end of their arms. We place a small triangular inner polygon completely contained in each overlap area of two such T-polygons. A variable gadget has exactly two minimum link fencings, which group the four pairs of consecutive inner polygons, either starting at an even or an odd position (see Fig. 6a and 6b). Each fence is actually a triangle and we spend 3/2 fence segments per inner variable polygon.

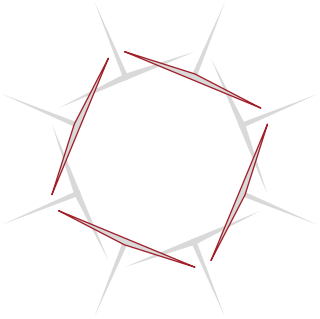
Wire gadget. A wire gadget $\mathcal{G}(v, c)$ is constructed for each occurrence of a variable v in a clause c. Its outer polygon is the union of a chain of an even number 2γ of overlapping triangles, as shown in Fig. 6d. An odd number of $2\gamma - 1$ inner triangles are placed in the area of the overlap between two consecutive outer triangles. The first triangle of $\mathcal{G}(v, c)$ overlaps with a true spike of $\mathcal{G}(v)$ if v occurs as a positive literal in c and with a false spike of $\mathcal{G}(v)$ otherwise (see Fig. 6c). We place one more inner triangle in this overlap area, yielding a total of 2γ inner triangles for $\mathcal{G}(v, c)$.

As for the variable gadget we can show that enclosing pairs of consecutive inner polygons of the wire gadget by triangular fences yields, locally, a minimum link fencing. If $P_1^{v,c}$ — the inner polygon in the overlap with $\mathcal{G}(v)$ — is fenced together with the next inner polygon $P_2^{v,c}$ of $\mathcal{G}(v, c)$ then the fencing of $\mathcal{G}(v, c)$ transmits *false*. The total number of fence segments is 3γ. Otherwise, if we fence $P_2^{v,c}$ together with $P_3^{v,c}$, the fencing transmits *true*. In this case, $P_{2\gamma}^{v,c}$ is fenced alone for a total of again 3γ fence links. It remains to fence $P_1^{v,c}$. If the state of $\mathcal{G}(v)$ corresponds to the true state of our literal, we can enclose $P_1^{v,c}$ in a fence of $\mathcal{G}(v)$ with two additional links, which will be charged to the clause c. If the state of $\mathcal{G}(v)$ corresponds to the false state of our literal, then fencing $P_1^{v,c}$ requires at least 3 additional links. Note that if the wire has state *false*, then there is no charge to c.

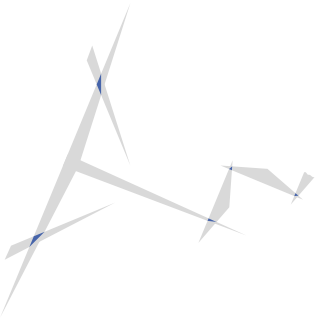
Clause gadget. For a clause $c = \{v_1, v_2, v_3\}$ the clause gadget $\mathcal{G}(c)$ is formed by an overlap of the last triangles of the three wires $\mathcal{G}(v_1, c)$, $\mathcal{G}(v_2 c)$, and $\mathcal{G}(v_3, c)$, see Fig. 8. This overlap



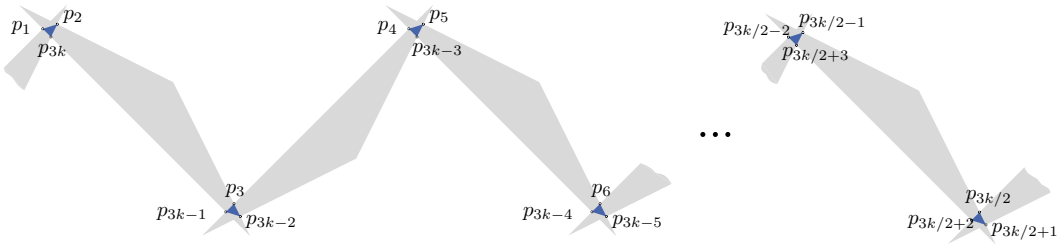
(a) $\mathcal{G}(v)$ in true state



(b) $\mathcal{G}(v)$ in false state

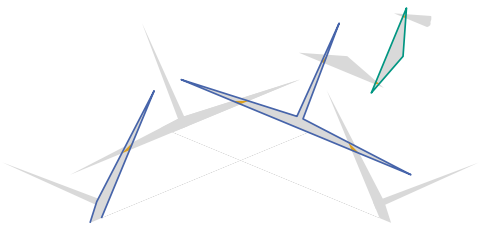


(c) Connection to $\mathcal{G}(v, c)$

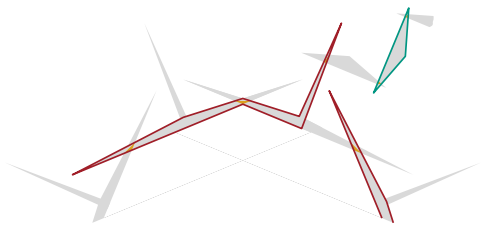


(d) Wire gadget $\mathcal{G}(v, c)$ with inner polygons

Figure 6 The two optimal fences of $\mathcal{G}(v)$ induce the true (a) and false (b) configuration of $\mathcal{G}(v)$. A connection between $\mathcal{G}(v)$ and $\mathcal{G}(v, c)$ is shown in (c).

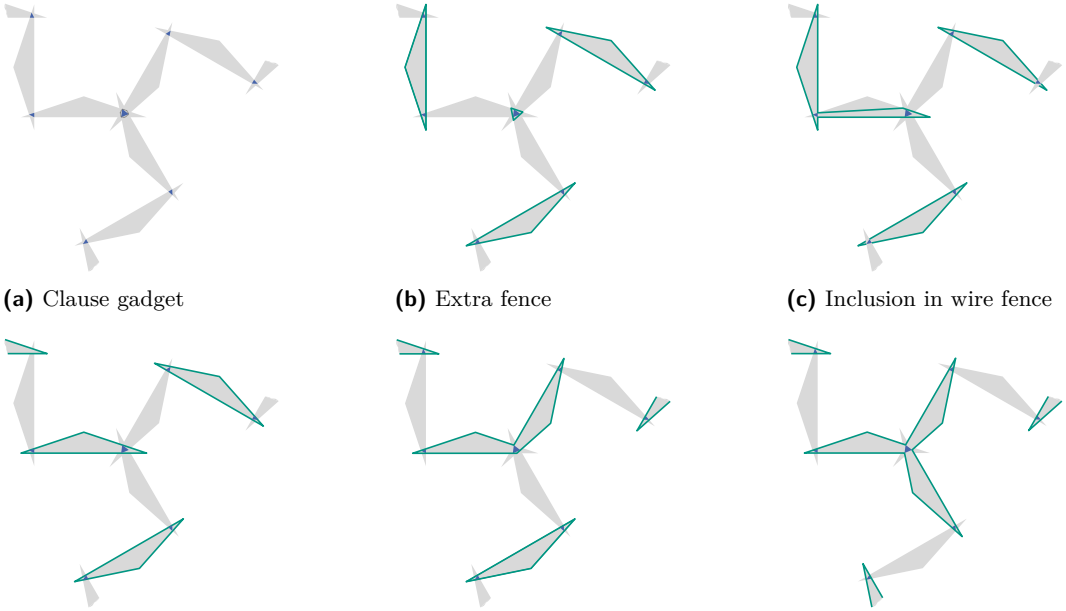


(a) $\mathcal{G}(v)$ in true state



(b) $\mathcal{G}(v)$ in false state

Figure 7 If $\mathcal{G}(v)$ is in the correct truth state (a) inclusion of the first inner polygon of $\mathcal{G}(v, c)$ induces an additional cost of two links , otherwise (b) the additional cost would clearly be higher.



(a) Clause gadget **(b)** Extra fence **(c)** Inclusion in wire fence

(d) Satisfied clause gadget **(e)** Two true wires **(f)** All wires true

Figure 8 The construction of a clause gadget $\mathcal{G}(c)$ (a) is such that three wires transmitting false would force the inner polygon of $\mathcal{G}(c)$ to be either enclosed by its own fence (b), or sub-optimally included in a fence of the wire gadget (c), both creating the need for additional links. If one wire transmits true, no additional links are needed. If more than one wire transmits true (e-f), the same number of links is required even if we include more than one last inner polygon in the fence of $\mathcal{G}(c)$.

area of the three triangles is filled with one inner polygon P_c, which forms the actual clause gadget. If one of the wires transmits *true*, then its last inner polygon P' is fenced by an individual fence and P_c can be included in the fence without additional cost, as they are contained in a bounding triangle. However, recall that we had to charge two fence segments to c in order to put this wire in its correct true state; otherwise the charge would even be three segments. If none of the three wires is in state *true*, then we must again spend at least three additional fence segments to enclose P_c, which we charge to such an unsatisfied clause c. Notice that if a variable assignment satisfies more than one literal of c, a minimum-link solution would still select only a single literal whose wire is put to the true state at a charge of two fence segments. If the assignment satisfies none of the three literals, then the resulting charge of c is at least three fence segments.

Correctness. It remains to show that ϕ is satisfiable if and only if the instance constructed above admits a fencing with at most $t = t(\phi)$ links. The value t is obtained from fencing all polygons in the variable and wire gadgets by triangular fences as desired and adding two more links for each satisfied clause. By construction this number of links is achievable if and only if each clause can be assigned one satisfying literal that is consistent with all other occurrences of the variable, the correctness of the (polynomial) reduction follows.

3 An algorithm for two-colored CMLF

In this section we present an algorithm for solving CMLF. Computing a minimum-link fence in this setting can be done by computing a fence for the convex hull of the contained polygons

with the algorithm by Wang [10] which runs in time $O(n \log n)$ with n being the number of corners of the contained polygons. Throughout this section an instance of CMLF is given as $(\mathcal{P}, Q)$ where Q is the outer polygon and $\mathcal{P}$ is the set of polygons contained in Q.

▶ **Lemma 3.1.** *Given an instance $(\mathcal{P}, Q)$ of CMLF, let $\mathcal{F}$ be a solution for $(\mathcal{P}, Q)$. There exists a solution $\mathcal{F}'$ for the CMLF instance $(CH(\mathcal{P}), Q)$ with $|F| = |F'|$.*

Proof. As F is a minimum-link fencing of $(\mathcal{P}, Q)$, it suffices to consider the case that for some minimum-link fencing $\mathcal{F}'$ of $(CH(\mathcal{P}), Q)$ we find $|F'| > |F|$. We will construct a new fence F° from this instance. Let $(p_1, \ldots, p_z)$ be the intersection points between F and $CH(\mathcal{P})$ ordered as they appear in a clockwise traversal of the convex hull, and observe that z is even. Let p_i, p_{i+1} be pairs of intersection points between F and $CH(\mathcal{P})$ such that the straight-line segment s_i connecting p_i and p_{i+1} lies on $CH(\mathcal{P})$ and completely outside of F (see Fig. 9). Consider the supporting line ℓ_i of s_i. If the fence lies completely in one of the closed half-planes bounded by ℓ_i we add s_i to F°.

Assume this is not the case. As s_i is on $CH(\mathcal{P})$ we get that ℓ_i does not intersect any polygon in $\mathcal{P}$. Moreover, as $\mathcal{F}$ consists of closed simple polygons we find two intersection points p'_i and p'_{i+1} that lie on ℓ_i, s.t., the parts of F appearing in a counterclockwise traversal from p_i to p'_i, as well as the ones in a clockwise traversal from p_{i+1} to p'_{i+1} lie outside of $CH(\mathcal{P})$. We add the segment s'_i between p'_i and p'_{i+1} to F°. Doing this for every pair of intersections we obtain a set of segments F°, where all segments are on the convex-hull of $\mathcal{P}$. Note that it is possible for these segments to intersect; if that is the case we only keep the parts until their intersection point. Finally, the start and end-points of connected chains of segments in F° lie on segments of polygons in $\mathcal{F}$. We can convert F° into a fence of $CH(\mathcal{P})$ by connecting these endpoints along the polygons in $\mathcal{F}$ and that fence will be disjoint from $\mathcal{P}$ (except possibly touching $\mathcal{P}$ in corner points).

It remains to argue that indeed $|F^\circ| \leq |F|$. We partition F° into two categories, segments that coincide with segments in F and segments that do not. Each of them is either a full segment of F or originates from the intersection of at most two different s'_i's and a segment of F. Furthermore, we add $z/2$ segments s'_i that are not sub-segments of segments in F. For each such s'_i we find at least one segment of F for which we did not add any sub-segment to F°. These are the segments of F on which p_i and p_{i+1} lie or that are fully outside of F°. ◀

▶ **Theorem 3.2.** *CMLF can be solved in time $O(n \log n)$ where n is the number of corners of polygons in $\mathcal{P}$.*

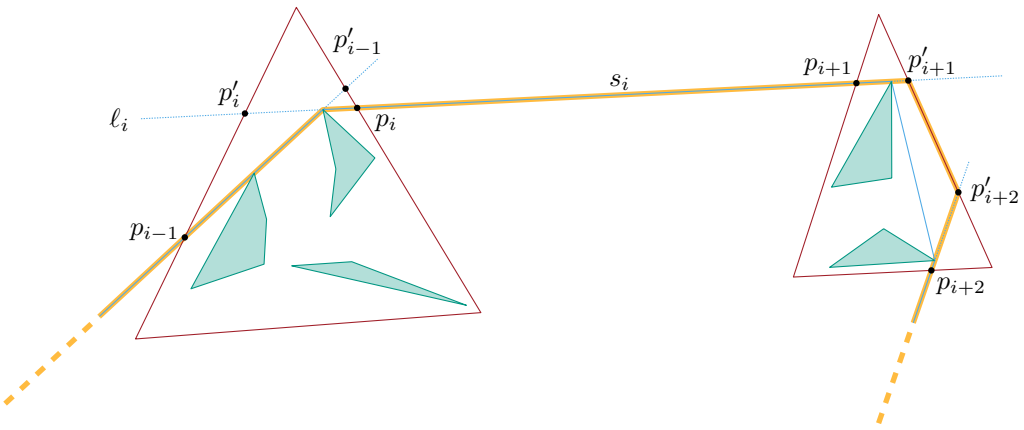


Figure 9 Computing a new fence (orange) from the old fences (purple) and the convex hull (blue).

References

1 Mikkel Abrahamsen, Anna Adamaszek, Karl Bringmann, Vincent Cohen-Addad, Mehran Mehr, Eva Rotenberg, Alan Roytman, and Mikkel Thorup. Fast fencing. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2018)*, pages 564–573, 2018. doi:10.1145/3188745.3188878.

2 Mikkel Abrahamsen, Panos Giannopoulos, Maarten Löffler, and Günter Rote. Geometric multicut. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *LIPIcs*, pages 9:1–9:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ICALP.2019.9.

3 Alok Aggarwal, Heather Booth, Joseph O'Rourke, Subhash Suri, and Chee K. Yap. Finding minimal convex nested polygons. *Information and Computation*, 83(1):98 – 110, 1989. doi:10.1016/0890-5401(89)90049-7.

4 Alok Aggarwal, Heather Booth, Joseph O'Rourke, Subhash Suri, and Chee-Keng Yap. Finding minimal convex nested polygons. *Information and Computation*, 83(1):98–110, 1989. doi:10.1016/0890-5401(89)90049-7.

5 Gautam Das. *Approximation schemes in computational geometry.* PhD thesis, University of Wisconsin, Madison, 1991.

6 Peter Eades and David Rappaport. The complexity of computing minimum separating polygons. *Pattern Recognition Letters*, 14(9):715–718, 1993. doi:10.1016/0167-8655(93)90140-9.

7 Subir Kumar Ghosh. Computing the visibility polygon from a convex set and related problems. *Journal of Algorithms*, 12(1):75–95, 1991. doi:10.1016/0196-6774(91)90024-S.

8 Klaus Jansen and Haiko Müller. The minimum broadcast time problem for several processor networks. *Theoretical Computer Science*, 147(1-2):69–85, 1995. doi:10.1016/0304-3975(94)00230-G.

9 Joseph SB Mitchell and Subhash Suri. Separation and approximation of polyhedral objects. *Computational Geometry: Theory and Applications*, 5(2):95–114, 1995. doi:10.1016/0925-7721(95)00006-U.

10 Cao An Wang. Finding minimal nested polygons. *BIT Computer Science section*, 31(2):230–236, 1991. doi:10.1007/BF01931283.

Characterizing Universal Reconfigurability of Modular Pivoting Robots*

Hugo A. Akitaya[1], Erik D. Demaine[2], Andrei Gonczi[3], Dylan H. Hendrickson[2], Adam Hesterberg[4], Matias Korman[5], Oliver Korten[6], Jayson Lynch[7], Irene Parada[8], and Vera Sacristán[†9]

1 **University of Massachusetts Lowell, USA**
hugo_akitaya@uml.edu
2 **Massachusetts Institute of Technology, USA**
{edemaine,dylanhen}@mit.edu
3 **Tufts University, USA**
andrei.gonczi@tufts.edu
4 **Harvard University, USA**
achesterberg@gmail.com
5 **Siemens Electronic Design Automation, USA**
matias_korman@mentor.com
6 **Columbia University, USA**
oliver.korten@columbia.edu
7 **University of Waterloo, Canada**
jayson.lynch@uwaterloo.ca
8 **TU Eindhoven, The Netherlands**
i.m.de.parada.munoz@tue.nl
9 **Universitat Politècnica de Catalunya, Spain**
vera.sacristan@upc.edu

Abstract

We give both efficient algorithms and hardness results for reconfiguring between two connected configurations of modules in the hexagonal grid. The reconfiguration moves that we consider are "pivots", where a hexagonal module rotates around a vertex shared with another module. Following prior work on modular robots, we define two natural sets of hexagon pivoting moves of increasing power: restricted and monkey moves. When we allow both moves, we present the first universal reconfiguration algorithm, which transforms between any two connected configurations using $O(n^3)$ monkey moves. On the other hand, if we only allow restricted moves or modules have a square shape, we prove that the reconfiguration problem becomes PSPACE-complete.

Related Version A full version of the paper is available on arXiv [2].

1 Introduction

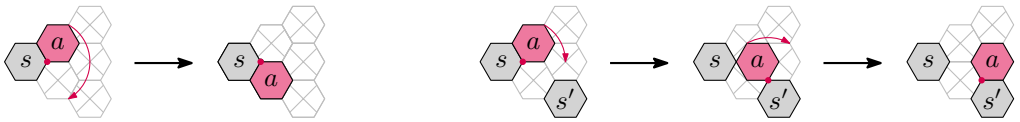
Reconfiguration problems encompass a large family of problems in which we need to provide a sequence of steps to transform one object into another. In this paper we consider the problem of reconfiguring a collection of modules in a lattice using some prespecified moves. In particular, we study the reconfiguration problem for edge-connected configurations of pivoting hexagons and squares. We require that edge-connectivity is maintained at all times (fulfilling the so-called **single backbone condition** [7]). The moves allowed are pivots: a

* This research started at the 34th Bellairs Winter Workshop on Computational Geometry in 2019. We want to thank all participants for the fruitful discussions and a stimulating environment.
† Partially supported by MTM2015-63791-R (MINECO/FEDER) and Gen. Cat. DGR 2017SGR1640.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

module can rotate around vertices shared with other modules and at the end of a move the pivoting module must lie in a lattice cell. The interior of two modules can never intersect.

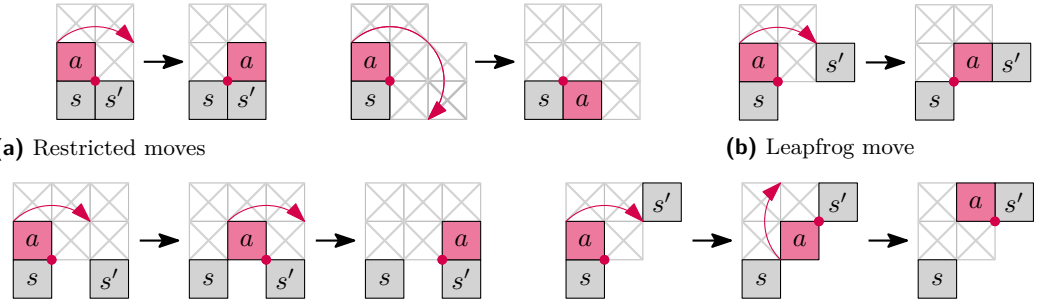
An hexagon can perform only two types of pivoting moves, illustrated in Figure 1. In a **restricted** move a module a adjacent to a module s pivots around a vertex v shared by a and s and ends the pivoting move in the other cell that has v on the boundary. The **restricted model** of pivoting only allows this move. In a **monkey** move a module a adjacent to a module s starts pivoting around a vertex v shared by a and s as in the restricted move, but halfway through the rotation another vertex w of a coincides with the vertex of a module s'. Then a continues the move pivoting around w in the same direction (clockwise or counterclockwise) as before until reaching a cell adjacent to s'. The **monkey model** of pivoting includes both the restricted move and the monkey move. Informally, the monkey move allows a module to keep pivoting in the same direction when a restricted move is not possible. Corners may only pivot on corners of other modules.



(a) Restricted move **(b)** Monkey move

Figure 1 Pivoting moves for hexagonal modules and the free cells required.

In the square grid two modules that share a vertex might not share an edge. Thus, for square modules there is a greater variety of pivoting moves. The **restricted**, **leapfrog**, and **monkey** moves are illustrated in Figure 2. The **restricted model** includes only restricted moves, the **leapfrog model** includes both restricted and leapfrog moves, and the **monkey model** includes all moves.



(a) Restricted moves **(b)** Leapfrog move

(c) Monkey moves

Figure 2 Pivoting moves for square modules and the free cells required.

Related work and contribution.

One of the most natural questions for modular robots is whether universal reconfiguration is possible. That is, is there an algorithm to transform any (connected) configuration of n modules into another configuration with the same number of modules?

Efficient algorithms are known for universal reconfiguration of modular robots using moves that have significantly lighter free-space requirements (fewer cells need to be empty to perform a move) [3, 6, 7, 8]. Relaxing the connectivity requirement has also lead to reconfigurability results [5].

The setting of this paper (pivoting modular robots) has proven to be more challenging. Instead, previous work has revolved around providing sufficient conditions for reconfiguration. Nguyen, Guibas and Kim [11] showed that reconfiguration of hexagonal modules using only restricted moves is always possible between configurations without the forbidden pattern illustrated in Figure 3 (left). Similarly, for pivoting squares, Sung et al. [12] presented an algorithm for reconfiguring between configurations without the patterns shown in Figure 3 (right). These algorithms fail to provide reconfiguration guarantees as soon as the configuration contains a single copy of the forbidden pattern. In an attempt to remove global requirements, a recent result [1] introduced a different type of necessary condition: they provide an efficient algorithm for reconfiguring between any two configurations that have 5 modules on the external boundary that can freely move (for pivoting squares in the monkey model). Other algorithms to reconfigure pivoting squares and hexagons are heuristics that do not provide termination guarantees [4, 10].

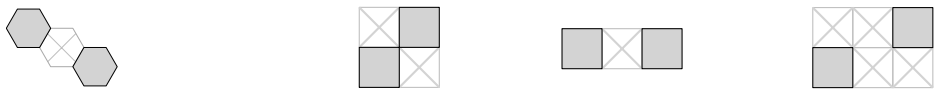


Figure 3 Forbidden patterns in previous algorithms for hexagonal and square pivoting modules

Despite many attempts, universal reconfiguration remains unsolved in the setting of edge-connected pivoting modules. In our work we answer this question for all five pivoting models for hexagons and squares. Specifically, we answer it positively for the hexagonal monkey model by giving a universal reconfiguration algorithm in Section 2. For all other models we show that it is PSPACE-hard to determine whether we can reconfigure from one configuration of modules into another one. A summarizing table of our results is shown in Table 1. Due to lack of space only a sketch of our algorithm is presented. Missing arguments can be found in the full version of this paper [2].

Model	Restricted	Leapfrog	Monkey
Hexagons	PSPACE-hard	N/A	$O(n^3)$ universal
Squares	PSPACE-hard	PSPACE-hard	PSPACE-hard $O(n^2)$ if +5 modules [1]

Table 1 Summary of results. The leapfrog moves do not make sense for hexagonal modules.

2 Polynomial algorithm for the hexagonal monkey model

Our approach uses a **canonical configuration** defined as the configuration with n modules whose contact graph is a path and each module is only adjacent to modules above and/or below it. Since each move is reversible, an algorithm that takes a configuration and transforms it into the canonical configuration within $O(n^3)$ moves can be used to compute $O(n^3)$ moves between any pair of configurations. The main strategy is to increase the connectivity of the contact graph[1]. Note that if the contact graph is 2-connected, every convex corner

[1] Increasing connectivity of the contact graph of the configuration is a concept that has recently proven useful in the different setting of reconfiguring **sliding squares** [9].

of the configuration is movable. Then, there is a module that can move to become the new topmost module by attaching itself to a previous topmost module. We then inductively building the canonical configuration.

Definitions and Preliminaries. The contact graph G is the adjacency graph of the modules in a configuration. Since connectivity is important for the problem, we use the **block tree** $\mathcal{B}$ of the contact graph G. See [2] for a definition. A graph is **2-connected** if it contains no cut vertices. A **block** (also 2-connected component) of G is a maximal subgraph of G that is 2-connected. The deletion of a cut vertex v of a connected graph G splits it into two or more components. A subgraph induced by such a component union with $\{v\}$ is called a **split component** of v. Similarly, a **2-cut** is a pair of vertices $\{v_1, v_2\}$ whose deletion increases the number of components of G. Its **2-split components** are the subgraphs induced by each of the components obtained by the deletion of $\{v_1, v_2\}$ union with $\{v_1, v_2\}$.

Let a **2-free** module be a module that is not in a 2-cut. A **crew** $c = (m_1, \ldots, m_k)$ is a sequence of modules that induce a connected component of G such that m_1 is 2-free, and m_i, $i \in \{2, \ldots, k\}$ is 2-free after the deletion of all m_j, $j \in \{1, \ldots, i-1\}$. For given 2-connected subgraph ℓ of G, let $\overline{\ell}$ be the induced subgraph of G given by $V(G) \setminus V(\ell)$. A **bridge** from ℓ is a crew that moved to create a path between ℓ and $\overline{\ell}$. One of the goals of the algorithm is to get a crew of size 3 in a group of grid positions called a **flower** that is otherwise empty. That allows us to maneuver the modules in the crew to create a bridge while not creating new blocks. Let a **flower** be a set of grid positions defined by a **center** cell and the six adjacent positions. A flower is **valid** for a 2-connected configuration ℓ and a disjoint crew c if it contains *exactly* the modules in c, and is adjacent to a module in ℓ.

A **row** containing a position p is the set of all positions $p + (-\frac{\sqrt{3}}{2}, \frac{1}{2})i$ for some integer i. An **extreme path** is a path induced by modules that are in the convex hull. Due to the geometry of the grid, extreme paths can only have six possible directions. A **SW extreme path** of a configuration ℓ is an ascending or descending path in the lower hull of ℓ.

Main algorithm. Here we give the general descriptions of our algorithm and the procedures it uses. The full description and analysis can be found in the full version [2]. We split the contact graph into two parts: the canonical path P which is a canonical configuration, and the remainder of the graph G. We initialize G to be the entire contact graph and P to be empty. Let $\mathcal{B}$ be the block tree of G rooted at the block containing the topmost rightmost module. We divide our algorithm into three phases.

Phase 1: Removing trivial leaves. This phase reconfigures a connected configuration into one without vertices of degree 1 (which are trivial leaves) in G. There are configurations in which it is not enough to just pivot the degree-1 modules, i.e., this task requires coordination with other modules. See Figure 4. From now, assume that every leaf of $\mathcal{B}$ contains at least 3 modules and no further procedure will change that.

▶ **Lemma 2.1.** *A connected configuration of $n > 2$ hexagons can be transformed in $O(n^2)$ moves into a configuration without trivial leaves in the contact graph without breaking connectivity.*

Phase 2: Merging leaves. The goal of this phase is to take a connected configuration with no degree 1 vertices, and transform it into a 2-connected configuration in $O(n^3)$ moves. The main technical tool of this phase is the procedure Merge which allows us to reduce the number of 2-connected components by merging them. Its input is a child (2-)split component of a cut vertex v (adjacent 2-cut $\{v_1, v_2\}$). We first apply the necessary rotations so that v

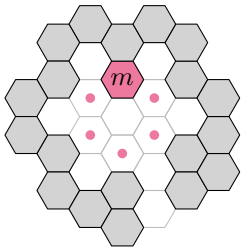


Figure 4 Configuration with one trivial leaf (m) that cannot be removed by pivoting it.

($\{v_1, v_2\}$) is farthest from the row ρ_0 containing the extreme SW path of ℓ. We then assume that ρ_0 does not include v ($\{v_1, v_2\}$) and neither does the row above it except for the base case when $|V(\ell)| = 3$ and ℓ is a split component, or when $|V(\ell)| = 5$ and ℓ is a 2-split component. The output is a set of modules that, after $O(|V(\ell)|^2)$ moves, bridges from ℓ.

We first identify a module m called the **ascending** module, defined as the topmost module in ρ_0. Note that m is movable. The overall strategy is to try to move m to its highest possible position in ρ_0 before it leaves ℓ. If m is 2-free, this can be done without affecting 2-connectivity; else, we make it 2-free using sub-procedures that either reduce the area enclosed by ℓ, or apply induction on a child 2-split component. We then make a bridge using m while it ascends in ρ_0 if it gets blocks by a vertex $m^* \notin \ell$, or accumulate 2-free modules at the top of the configuration where a valid flower will form. Then, a sub-procedure moves the valid flower around ℓ until it hits $\overline{\ell}$ where we create a bridge with the crew.

▶ **Lemma 2.2.** *If $\ell \neq G$ is a leaf block of $\mathcal{B}$, Merge(ℓ) performs $O(|V(\ell)|^2)$ moves merging ℓ and a subset of nodes of $\mathcal{B}$ into a single block while not creating any other new blocks.*

▶ **Corollary 2.3.** *G can be made 2-connected in $O(n^3)$ moves.*

Phase 3: Building the canonical path P. Once the configuration is 2-connected, we can start moving modules onto the end of our path P at a cost of $O(n^2)$ moves per module. We use a slightly modified version of Merge to produce a crew that can move to P without breaking the 2-connectivity of G.

▶ **Lemma 2.4.** *If G is 2-connected, in $O(n^2)$ moves we can produce a 2-free module on an extreme path of G while maintaining the 2-connectivity of G.*

Our main theorem is a direct consequence of Lemma 2.1, Corollary 2.3 and Lemma 2.4.

▶ **Theorem 2.5.** *Any connected configuration of n hexagonal modular robots can be reconfigured to any other with $O(n^3)$ pivoting moves in the monkey model, while maintaining connectivity.*

3 Conclusions

Although this paper settles the question of whether universal reconfiguration is possible for all models, it also spans several interesting problems. First, for hexagonal modules under the monkey model (where universal reconfiguration is possible), there is a gap between the upper bound of our algorithm (Theorem 2.5) and the naive $\Omega(n^2)$ lower bound (number of moves needed to transform a horizontal strip into a compact hexagon). Even if the gap is closed (possibly with a completely different algorithm), then the interest would be to design a distributed algorithm and/or to consider a strategy that does many moves in parallel.

For the other models universal reconfiguration is not possible, but it would be nice to find a local property that would allow reconfiguration between many configurations. For example, if we allow monkey moves with square modules, reconfiguration is possible as long as both configurations have five modules on the outer shell that can move [1]. We wonder if the concept of musketeers or crew can be extended to other models. We remark that for the hexagonal monkey model this technique cannot be directly applied. Indeed, a key step of that approach was that whenever there is no module that can freely pivot on the external boundary, we bridge such that a well-defined module m (extremal for a potential function defined on the coordinates of the modules) can pivot along the external boundary without disconnecting the configuration. Moreover, as many modules as helpers used for bridging can be liberated locally around m. However, this is not always possible for hexagonal modules, as shown in Figure 5: Three helper modules are necessary for bridging, but only two can be liberated locally around m.

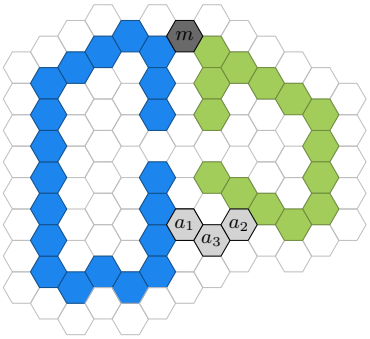


Figure 5 The strategy in [1] cannot be directly translated to pivoting hexagons in the monkey model.

References

1 Hugo A. Akitaya, Esther M. Arkin, Mirela Damian, Erik D. Demaine, Vida Dujmovićc, Robin Flatland, Matias Korman, Belén Palop, Irene Parada, André van Renssen, and Vera Sacristán. Universal reconfiguration of facet-connected modular robots by pivots: The $O(1)$ musketeers. In *Proc. 27th Annual European Symposium on Algorithms (ESA)*, volume 144, pages 3:1–3:14, 2019. doi:10.4230/LIPIcs.ESA.2019.3.

2 Hugo A. Akitaya, Erik D. Demaine, Andrei Gonczi, Dylan H. Hendrickson, Adam Hesterberg, Matias Korman, Oliver Korten, Jayson Lynch, Irene Parada, and Vera Sacristán. Characterizing universal reconfigurability of modular pivoting robots. *CoRR*, abs/2012.07556, 2020. arXiv:2012.07556.

3 Greg Aloupis, Sébastien Collette, Mirela Damian, Erik D. Demaine, Robin Flatland, Stefan Langerman, Joseph O'Rourke, Val Pinciu, Suneeta Ramaswami, Vera Sacristán, and Stefanie Wuhrer. Efficient constant-velocity reconfiguration of crystalline robots. *Robotica*, 29(1):59–71, 2011. doi:10.1017/S026357471000072X.

4 Nora Ayanian, Paul J. White, Ádám Hálász, Mark Yim, and Vijay Kumar. Stochastic control for self-assembly of XBots. In *Proc. ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC-CIE)*, pages 1169–1176, 2008. doi:10.1115/DETC2008-49535.

5 Nadia M. Benbernou. *Geometric Algorithms for Reconfigurable Structures*. PhD thesis, Massachusetts Institute of Technology, 2011.

6 Adrian Dumitrescu and János Pach. Pushing squares around. *Graphs and Combinatorics*, 22(1):37–50, 2006. doi:10.1007/s00373-005-0640-1.

7 Adrian Dumitrescu, Ichiro Suzuki, and Masafumi Yamashita. Motion planning for metamorphic systems: feasibility, decidability, and distributed reconfiguration. *IEEE Transactions on Robotics*, 20(3):409–418, 2004. doi:10.1109/TRA.2004.824936.

8 Robert Fitch, Zack Butler, and Daniela Rus. Reconfiguration planning for heterogeneous self-reconfiguring robots. In *Proc. 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2460–2467, 2003. doi:10.1109/IROS.2003.1249239.

9 Irina Kostitsyna, Irene Parada, Willem Sonke, Bettina Speckmann, and Jules Wulms. Compacting squares. Manuscript, 2020.

10 Tom Larkworthy and Subramanian Ramamoorthy. A characterization of the reconfiguration space of self-reconfiguring robotic systems. *Robotica*, 29(1):73–85, 2011. doi:10.1017/S0263574710000718.

11 An Nguyen, Leonidas J. Guibas, and Mark Yim. Controlled module density helps reconfiguration planning. In *Algorithmic and Computational Robotics: New Dimensions (2000 WAFR)*, pages 23–36. 2001.

12 Cynthia Sung, James Bern, John Romanishin, and Daniela Rus. Reconfiguration planning for pivoting cube modular robots. In *Proc. 2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1933–1940, 2015. doi:10.1109/ICRA.2015.7139451.

Rectangular Spiral Galaxies are Still Hard*

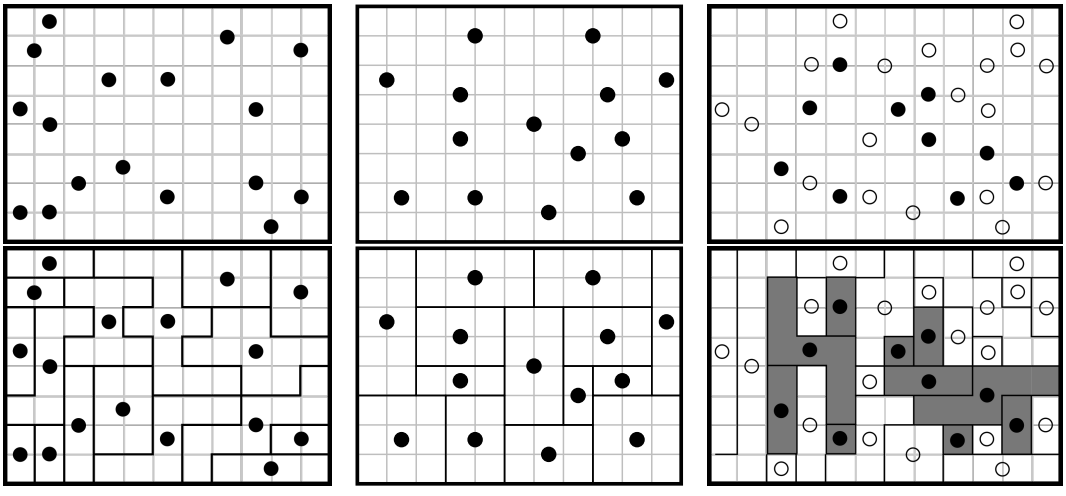
Erik D. Demaine[1], Maarten Löffler[2], and Christiane Schmidt[2]

1 **Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology**
`edemaine@mit.edu`
2 **Department of Information and Computing Sciences, Universiteit Utrecht**
`m.loffler@uu.nl`
3 **Communications and Transport Systems, ITN, Linköping University**
`christiane.schmidt@liu.se`

Abstract

Spiral Galaxies is a pencil-and-paper puzzle played on a grid of unit squares with a given set of points, referred to as *centers*. The grid needs to be partitioned into polyominoes such that each polyomino contains exactly one center and is 180° rotationally symmetric about its center. We show that the puzzle is NP-complete even if the polyominoes are restricted to be (a) rectangles of arbitrary size or (b) 1×1, 1×3, and 3×1 rectangles. The proof for the latter variant also implies that finding a non-crossing matching in modified grid graphs where edges connect vertices of distance 2 is NP-complete. Moreover, we prove that minimizing the number of centers, such that there exist a set of Spiral Galaxies that exactly cover a given shape, is NP-complete.

1 Introduction



(a) (b) (c)

Figure 1 Spiral Galaxies puzzles in several styles; solutions in the bottom row. (a) Classic Spiral Galaxies puzzle. (b) Rectangular Galaxies puzzle. (c) Spiral Galaxies puzzle with black and white centers such that the polyominoes containing the black centers in the solution yield a picture.

Spiral Galaxies is a pencil-and-paper puzzle published by Nikoli in 2001 [5] under the name "Tentai Show" [6]. It is played on a grid of unit squares with given *centers*: points

* This research was performed in part at the 30th and the 33rd Bellairs Winter Workshop on Computational Geometry. We thank all other participants for a fruitful atmosphere.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

that are located at grid vertices, square centers or edge midpoints. The goal is to decompose the grid into polyominoes, such that each polyomino contains exactly one center and is 180° rotationally symmetric about its center; see Figure 1 (a). The solution for a Spiral Galaxies puzzle may not be unique, but generally puzzles are designed to have a unique solution.

Friedman [4] showed Spiral Galaxies to be NP-complete for general polyomino shapes, and Fertin et al. [3] showed NP-hardness for Spiral Galaxies of size at most seven. In this paper we consider Rectangular Galaxies; a variant of Spiral Galaxies in which all polyominoes are required to be rectangles; see Figure 1 (b). We consider the general case and a special case in which we allow only 1×1, 1×3, and 3×1 rectangles. We show that both puzzle variants are (still) NP-complete. The proof for the latter puzzle also implies that finding a non-crossing matching in squared grid graphs (that is, modified grid graphs where edges connect vertices at distance 2) is NP-complete.

In another variant of the puzzle, a subset of the centers is colored black, and the polyominoes that contain these centers yield a picture as a solution [6]; see Figure 1 (c). Logic puzzles which yield pictures when solved are a popular genre; one famous example is the nonogram [7]. Such puzzles can also be used to construct fonts [1]. Constructing an interesting puzzle such that its solution is a given target shape is non-trivial: while a valid puzzle trivially exists, by simply placing a center in every grid cell, the resulting puzzle is clearly not interesting. We are, hence, also interested in finding the minimum number of centers, such that there exist spiral galaxies that exactly cover a given shape. We also prove this problem to be NP-complete.

1.1 Notation and Preliminaries

The game is played on an $m \times n$ grid. Centers are placed on the grid (on square centers, edge midpoints, or grid vertices), either all of them have the same color, or a subset of the centers may be colored black. To solve each puzzle, the solver is required to partition the board with polyominoes, such that each polyomino contains a single center and is 180° rotationally symmetric about its center. The grid and the centers form a Spiral Galaxies board B.

For our reductions, we use the PLANAR 1-IN-3-SAT PROBLEM, a well known NP-complete and $\#P$-complete problem [2].

2 Rectangular Galaxies

In this section, we show that the problem of solving Sprial Galaxies boards is NP-complete when all galaxies are restricted to be rectangles.

▶ **Theorem 1.** *Determining if a Spiral Galaxies board is solvable with only rectangular galaxies is NP-complete.*

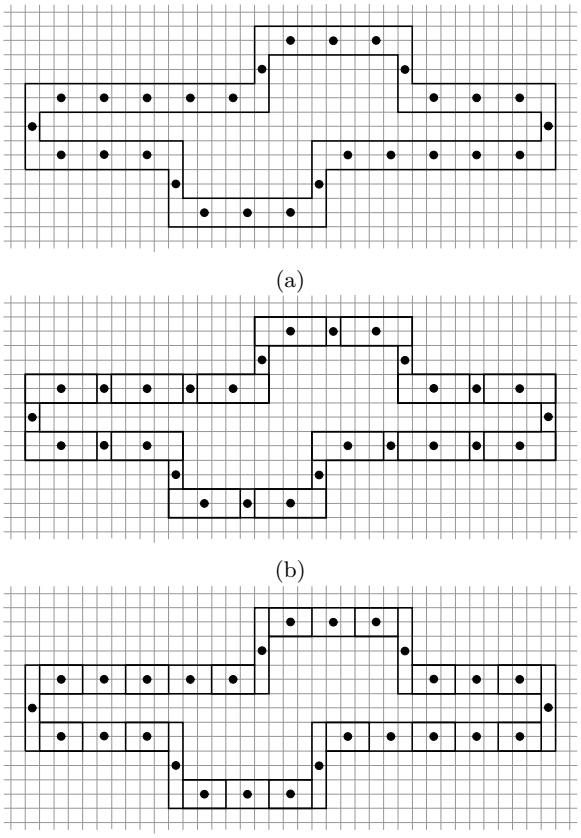
Proof. We give a reduction from PLANAR 1-IN-3-SAT. Given an instance F of PLANAR 1-IN-3-SAT with incidence graph G, we show how to turn a rectilinear planar embedding of G into a Spiral Galaxies board B such that a solution to B yields a solution to F.

We begin by constructing a representation of variables, we then show how to propagate and bend the variable values using 'wires'/'corridors' and combine the wires to form clauses.

The grid regions not incorporated into our gadgets will not affect the gadget solutions, as for any open region we use a **face gadget**, shown in Figure 2, to force the region to contain single square galaxies, which disallow any other Spiral Galaxies to occupy these grid squares. Therefore, there can be no interference between our gadgets and the grid regions surrounding

(a) (b)

Figure 2 Face gadget to fill in the space inbetween all other gadgets.



(a) (b) (c)

Figure 3 (a) Variable loop with two possible states (b) and (c) corresponding to a truth assignment of “true” and “false” of the corresponding variable.

them. Note that our face gadget forces us to not use open regions of width 1, as these would not enforce single square galaxies.

We construct **variable loops** as shown in Figure 3. They are constructed with centers located at edge centers within distance of 3 of each other, the loop is closed by additional centers at edge center points. Each variable loop has two possible solutions, corresponding to setting the variable as "true" or "false", and the rectangle placement is completely determined by the variable assignment.

From each variable loop, we can propagate the variable value, creating a **corridor gadget**, shown in Figure 4. The corridor gadgets are shown in blue and have a center distance of 5 on edge centers. Again, only two truth assignments are possible, each correspond to setting the variable to “true” (Figure 4(b)) or “false” (Figure 4(c)). If the corridor does not pick up the correct signal from the variable loop, the variable loop cannot be solved with rectangular

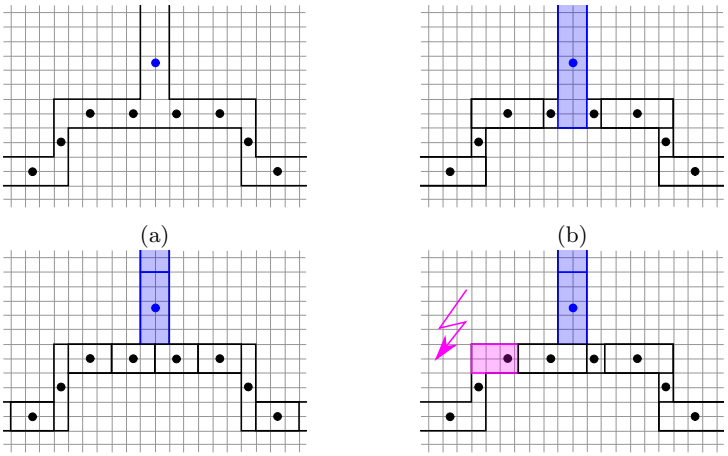


Figure 4 (a) Variable loops (black centers) and connecting variable corridors (blue centers). (b)/(c) Two possible truth assignments of the connecting corridor depending on the truth assignment in the variable loop. (d) If the corridor does not pick up the correct signal from the variable loop the variable loop cannot be completed.

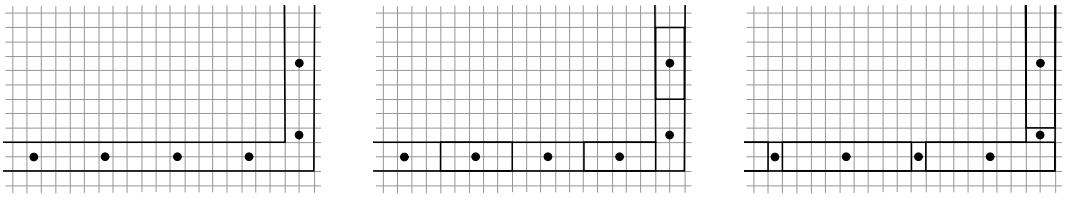


Figure 5 (a) Variable corridors with bend, (b)/(c) two possible covers with rectangular galaxies.

Spiral Galaxies, see, e.g., Figure 4(d). We show the **corridor bend** gadget in Figure 5. Moreover, we use a **corridor shift** gadget, shown in Figure 6, which allows us to shift the location of the corridor by any number larger than 3. These are used to connect to the clause gadgets in the appropriate places.

To combine the corridor gadgets into clauses, we use the **clause gadget** shown in Figure 7. At least one of the three variables' truth assignments is forced to be true (see Figure 7(b)-(d)), more than one cannot be set to true (see Figure 7(e)-(h)), hence, it forces exactly 1 true assignment, giving a solution to the instance F.

A solution to an $m \times n$ Spiral Galaxies board can be verified in polynomial time. ◀

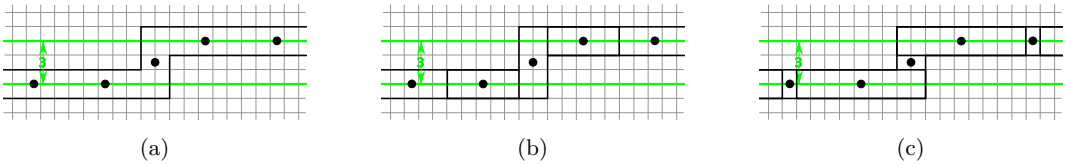


Figure 6 (a) Variable corridors can be shifted by any number larger than 3 (here shown for a shift of 3 as indicated in green). (b) and (c) depict the different variable assignments.

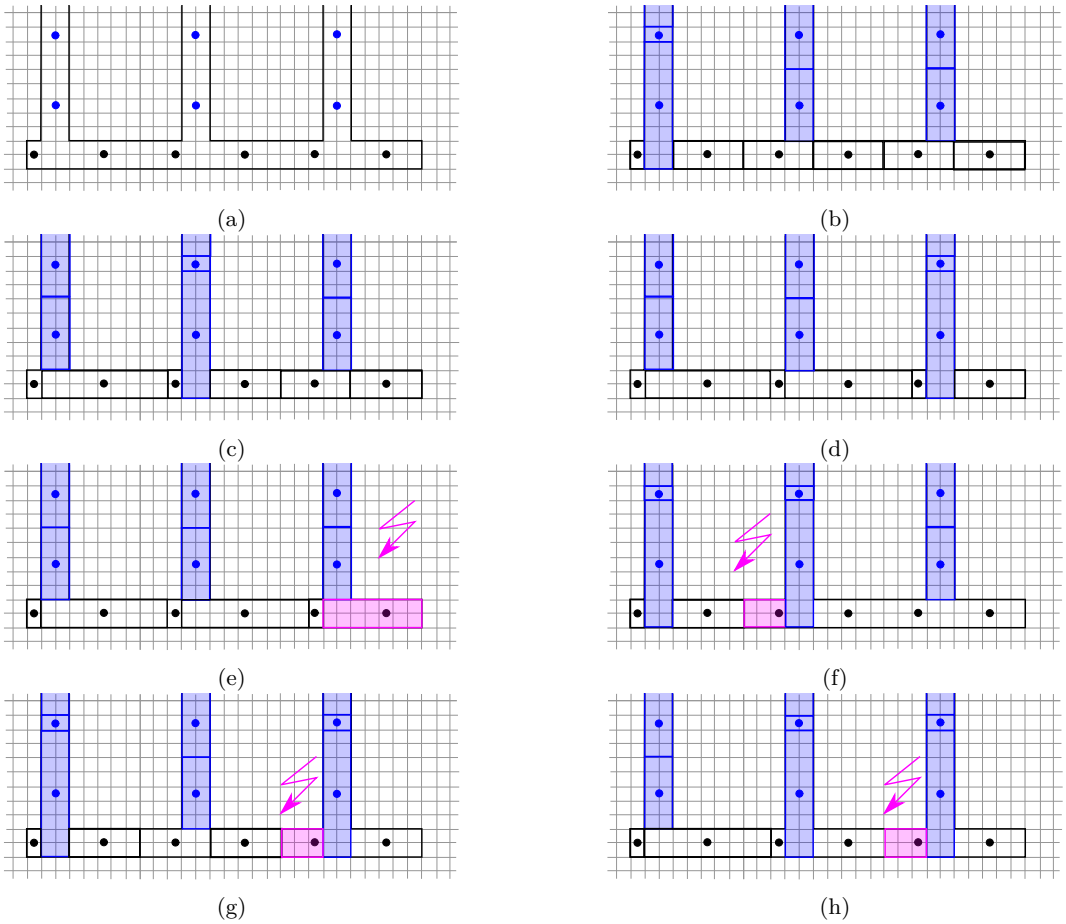


Figure 7 (a) Clause gadget shown in black and the connecting variable corridors shown in blue. (b)-(d) The clause gadget can be filled in with rectangle galaxies iff exactly one of the three connecting variables has a truth assignment that fulfills the clause. The clause cannot be completed if all variables do not fulfill the clause (e), or if more than one variable fulfills the clause (f)-(h).

3 Spiral Galaxies with 1×1, 1×3, and 3×1 Rectangles

In this section, we show that the problem of solving Spiral Galaxies boards is NP-complete, even when only $1 \times 1, 1 \times 3$ and 3×1 galaxies are allowed, and that the problem of counting the number of solutions to a Spiral Galaxies board with these galaxy types is $\#P$-complete and ASP-complete, see [8] Chapter 28 and [9, 10] for definitions of $\#P$-completeness and ASP-completeness, respectively. For our reductions, we—again—use the PLANAR 1-IN-3-SAT PROBLEM.

▶ **Theorem 2.** *Determining if a Spiral Galaxies board is solvable with only* $1 \times 1, 1 \times 3$ *and* 3×1 *galaxies is NP-complete and counting the number of solutions is #P-complete and ASP-complete.*

Proof. The proof is by reduction from PLANAR 1-IN-3-SAT. Given an instance F of planar 1-in-3-SAT with incidence graph G, we show how to turn a rectilinear planar embedding of G into a Spiral Galaxies board B such that a solution to B yields a solution to F, thereby showing NP-completeness. Furthermore, there will be a one-to-one correspondence between solutions of B and solutions of F, showing #P-completeness and ASP-completeness.

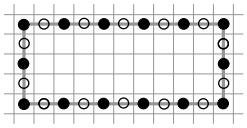
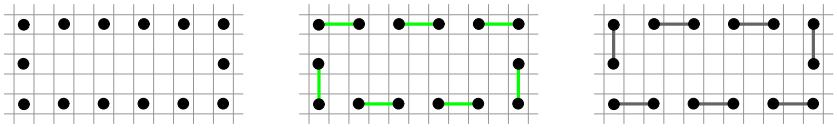


Figure 8 We place centers (black circles) on the middle point of potential edges (light gray) between any pair of black disks. In the remainder of this Section, we show only the black disks, but not the centers.



(a) (b) (c)

Figure 9 (a) Variable loop with two possible states (b) and (c) corresponding to a truth assignment of "true" and "false" of the corresponding variable.

In this proof, disks in the figures will not show centers for Spiral Galaxies. Disks with distance 2 can be connected by an edge, centers will be located at the middle of each potential edge, see Figure 8. Hence, any 1×3 or 3×1 galaxy will cover both disks, denoted by an edge between these two disks; any 1×1 galaxy will not extend over the disks, and is shown by a non-existing edge between the disks.

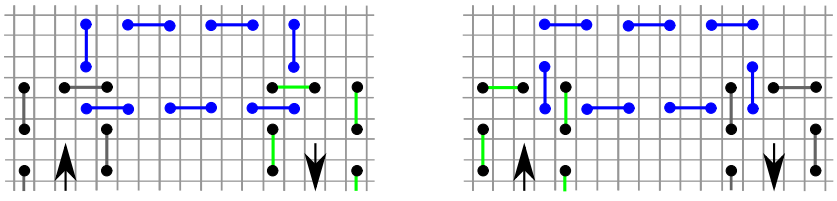
We start with constructing variable gadgets that represent variables and their negations, we show how we can negate the truth assignment of a variable, and construct clause gadgets in which we combine the variable loops. Throughout the discussion, the constructed gadgets have a single solution for any given variable assignment, which will make this reduction parsimonious.

The board region not incorporated into our gadgets will be filled with centers at square centers which will force the region to be filled with unit square galaxies, and will disallow any other type of galaxies. Thus, there is no interference between our gadgets and the board area surrouding them. Again, the face gadget forces us to not use open regions of width 1, as these would not enforce single square galaxies.

The **variable loop**, shown in Figure 11, has two possible solutions (shown in Figure 11(b) and (c)), each corresponding to one truth assignment for the variable ("true" and "false"). We extend the size of the variable loop to connect to other gadgets—to build **corridors**.

Negating a variable corresponds to inserting a **negation gadget** into the corridor, and to continue with another variable corridor as in Figure 10.

We combine the variable corridors into a **clause gadget**, see Figure 11: the variable



(a) (b)

Figure 10 Negation gadget in blue, with two black variable loops. The incoming loop on the left, has a different truth assignment than the outgoing loop on the right, two cases shown in (a)/(b).

loops are shown in black, and the clause gadget in gray. There are three possible states of the clause gadget, depicted in red, turquoise, and violet (see Figure 11(b)). Each state forces exactly one of the variables' truth assignments to fulfill the clause assignment (all cases are shown in Figure 11(c)-(e)), giving a solution to the instance F.

A solution to an $m \times n$ Spiral Galaxies board can be verified in polynomial time. ◀

4 Non-Crossing Matching in Squared Grid Graphs

We define a *squared grid graph* as a modified grid graph where edges connect vertices at distance 2. From the proof of Theorem 3.1—interpreting the disks as graph vertices—we obtain a corollary:

▶ **Corollary 3.** *Non-crossing matching in squared grid graphs is NP-complete.*

5 Minimizing Centers in Spiral Galaxies for a Given Shape

In this section, we are given a (black) shape $\mathcal{S}$ on a Spiral Galaxies board, and we aim to find the minimum number of centers, such that there exist spiral galaxies with these centers that exactly covers the given shape $\mathcal{S}$. We show that this problem is NP-complete by a reduction—one more time—from PLANAR 1-IN-3-SAT.

▶ **Theorem 4.** *Minimizing the number of centers on a Spiral Galaxies board, such that Spiral Galaxies with these centers exactly cover a given shape $\mathcal{S}$ is NP-complete.*

Proof. First, we introduce *local center gadgets*: thin constructions with unique shapes, which ensure that there must be at least one center in each of them; see Figure 12. The idea will now be to construct a shape which can be covered with exactly this set of centers if and only if the 1-in-3-sat instance is satisfiable.

To this end, we create *block gadgets*: 5×5 rooms which are connected to local center gadgets on two or three sides. We distinquish *straight blocks*, *corner blocks*, and *clause blocks*; see Figure 13. Next, we define the *fix gadget*: an alternating sequence of four local center gadgets and three block gadgets making a U-turn as in Figure 14.

We claim that in any chain of blocks that contains a fix gadget, each block must completely belong to a single galaxy. Indeed, suppose we have a fix gadget consisting of blocks A, B, and C, where A and C are corner blocks but B is a straight block, and suppose A is split among multiple galaxies, say a red one at the top and a blue one at the right (refer to Figure 14 (b)). Then the bottom center pixel of B must also be red, and hence the top center pixel of C must be red by summetry. But then C must be completely red; contradiction.

A **variable chain** consists of alternating local center gadgets and block gadgets starting and ending at *end gadgets*; see Figure 16 for a variable chain and Figure 15 for an end gadget. Each variable chain must include a fix gadget.

A clause is simply a single clause block where three variable chains meet: it can be solved without using an additional center if and only if precisely one of the three chains needs to use the block. The only missing ingredient now is a *split gadget*, which ensures we can make multiple variable chains with the same state. We let several variable chains end in a common area, which, similar to the end gadget, can be solved with one center only if all blocks are present, or if all blocks are absent; see Figure 17. Note that the distance between adjacent block gadgets may be adjusted as needed; this also allows us to negate variables. ◀

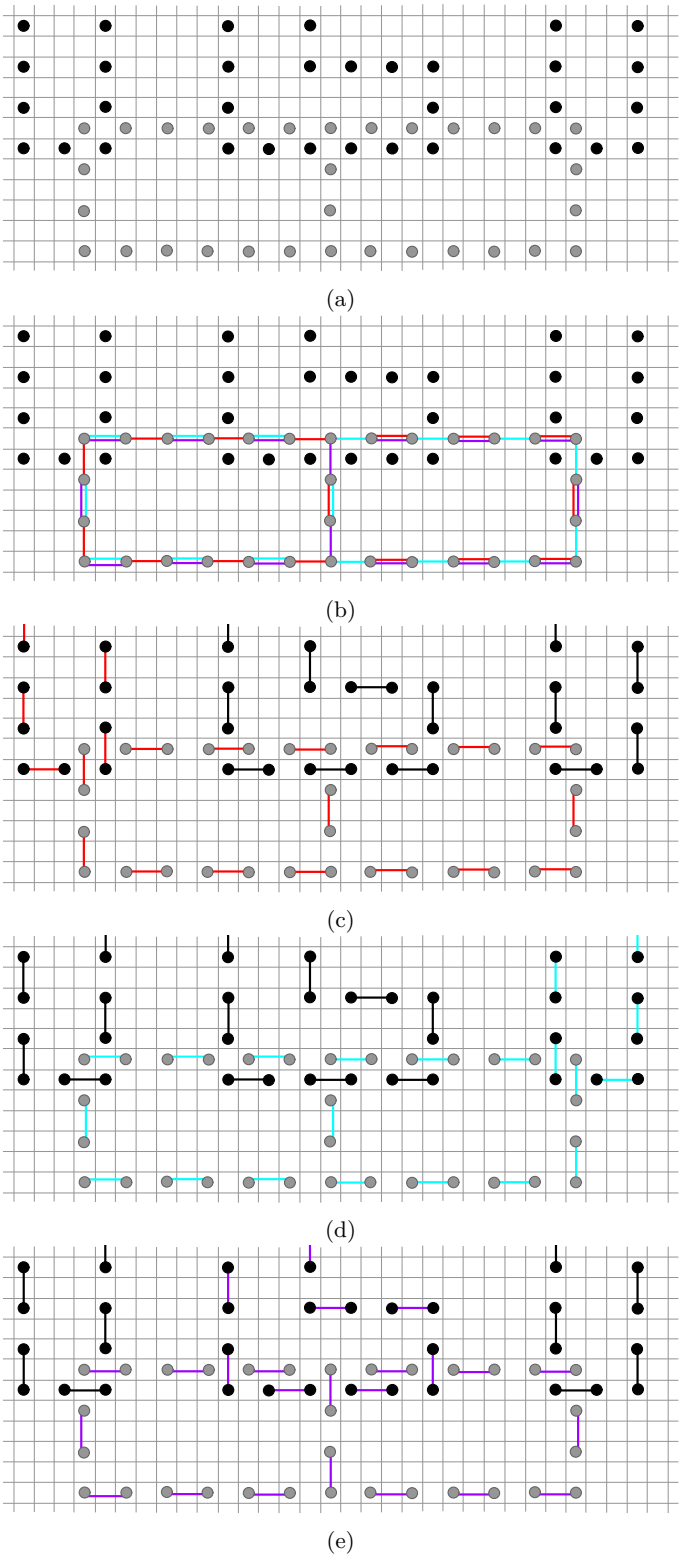


Figure 11 (a) Clause gadget, in gray, with the three incoming variable loops, in black. (b) shows the three possible states of the clause gadget, and (c)-(e) give each one assignment in the clause gadget, with the corresponding assignments of the variable loops, the (only) variable with a truth assignments that fulfills the clause is shown in the same color as the clause edges.

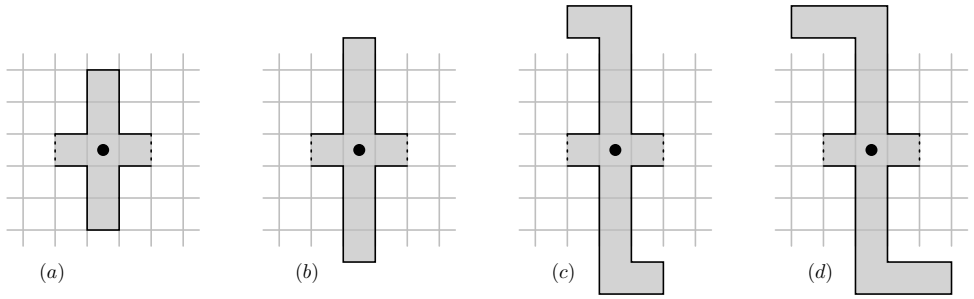


Figure 12 The local center gadget must have at least one center. (a-d) Different shapes ensure we cannot include them in larger galaxies.

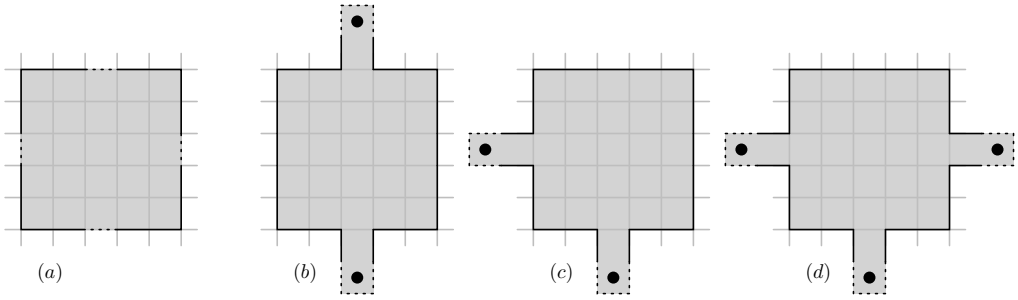


Figure 13 (a) The block gadget. (b) A straight block. (c) A corner block. (d) A clause block.

6 Some Spiral Galaxies Puzzles

The Spiral Galaxies board from Figure 18 solves for the black letters A, B, H, R, S, Z (and for disconnected galaxies also for the letter E). See Figures 19 and 20 for solutions.

References

1 Walker Anderson, Erik D. Demaine, and Martin L. Demaine. Spiral Galaxies font. *The Mathematics of Various Entertaining Subjects (MOVES 2017)*, 3:24–30, 2019. See also `http://erikdemaine.org/fonts/spiralgalaxies/`.

2 Martin E. Dyer and Alan M. Frieze. Planar 3DM is NP-complete. *Journal of Algorithms*, 7(2):174–184, 1986.

3 Guillaume Fertin, Shahrad Jamshidi, and Christian Komusiewicz. Towards an algorithmic guide to Spiral Galaxies. *Theoretical Computer Science*, pages 26–39, 2015.

4 Erich Friedman. Spiral galaxies puzzles are NP-complete. URL: `https://erich-friedman.github.io/papers/spiral.pdf`, March 2002.

5 Nikoli. Tentai Show. URL:`http://nikoli.co.jp/en/puzzles/astronomical_show.html`.

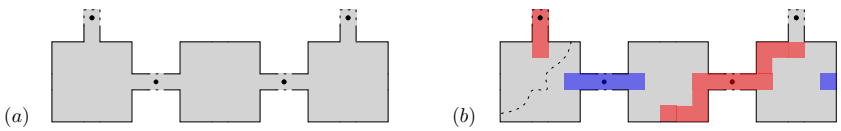


Figure 14 (a) The fix gadget. (b) If the left block is split among multiple galaxies: contradiction.

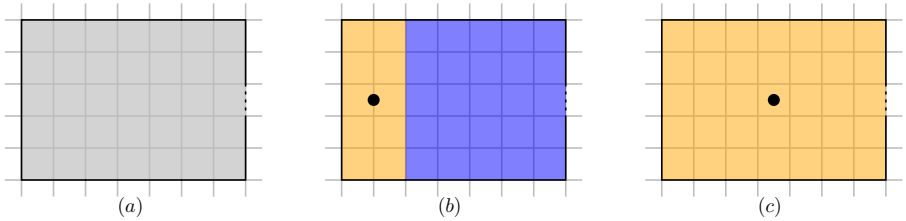


Figure 15 An end gadget has one center, whether or not a block is used by the galaxy next to it.

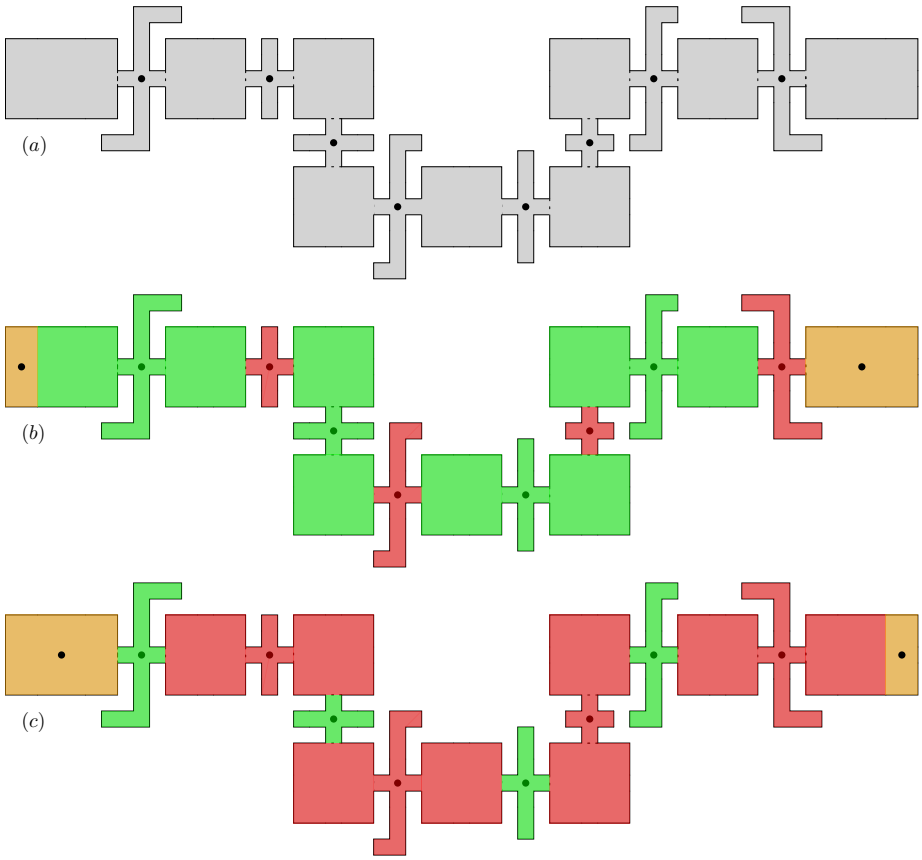


Figure 16 A variable chain and its two possible truth assignments.

6 Grandmaster Puzzles. Spiral Galaxies. URL: `https://www.gmpuzzles.com/blog/spiral-galaxies-rules-info/`. accessed April 4, 2018.

7 Nobuhisa Ueda and Tadaaki Nagao. NP-completeness results for NONOGRAM via parsimonious reductions. Technical Report TR96-0008, Department of Computer Science, Tokyo Institute of Technology, Tokyo, Japan, May 1996.

8 Vijay V. Vazirani. *Approximation Algorithms*. Springer Publishing Company, Incorporated, 2010.

9 Takayuki Yato. Complexity and completeness of finding another solution and its application to puzzles, 2003.

10 Takayuki Yato and Takahiro Seta. Complexity and completeness of finding another solution and its application to puzzles. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 86(5):1052–1060, 2003.

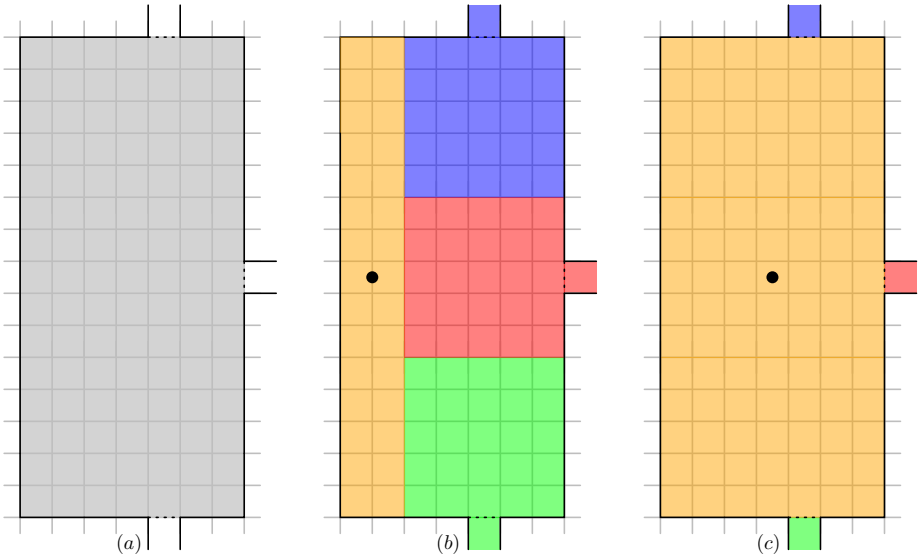


Figure 17 The split gadget is essentially three end gadgets, but can only be solved with a single center if either all three blocks are used or all three are not used.

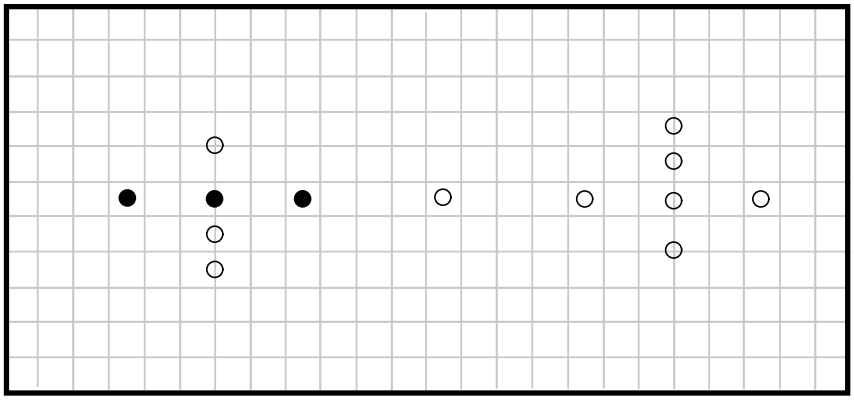
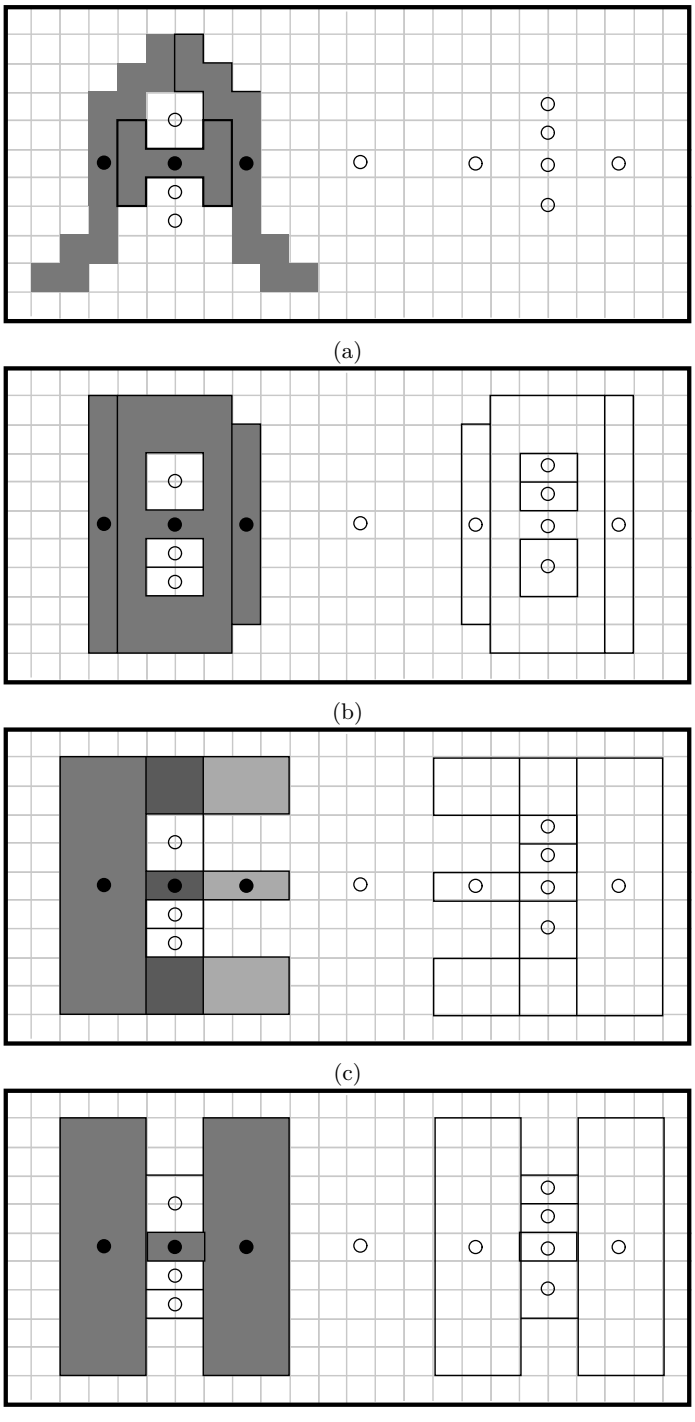


Figure 18 Puzzle that can be solved for letters A, B, H, P, R, S, Z (E for disconnected galaxies).



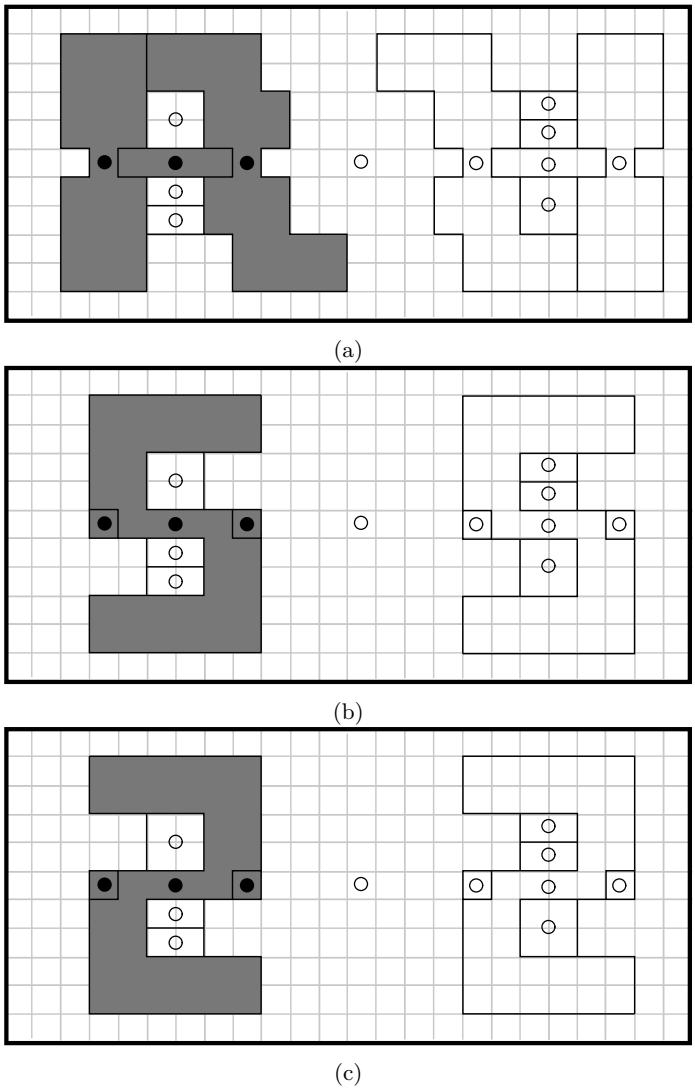
(a)

(b)

(c)

(d)

Figure 19 Puzzle solutions for letters A, B, E, H.



(a)

(b)

(c)

Figure 20 Puzzle solutions for letters R, S, Z.

Deletion only Dynamic Connectivity for Disk Graphs*

Haim Kaplan[1], Katharina Klost[2], Kristin Knorr[†2], Wolfgang Mulzer[‡2], and Liam Roditty[3]

1 **School of Computer Science, Tel Aviv University**
haimk@tau.ac.il
2 **Institut für Informatik, Freie Universität Berlin**
{kathklost, knorrkri, mulzer}@inf.fu-berlin.de
3 **Department of Computer Science, Bar Ilan University**
liamr@macs.biu.ac.il

Abstract

Let $S \subset \mathbb{R}^2$ be a set of n sites where $s \in S$ has an associated radius r_s defining a disk D_s. Then the *disk graph* $D(S)$ has a vertex for every site and two sites s, t are connected by an undirected edge if and only if $D_s \cap D_t \neq \emptyset$. We present a data structure which serves a sequence of n deletions and k connectivity queries in time $O(n \operatorname{polylog} n + k(\log n / \log \log n))$.

1 Introduction

The question if two vertices in a given graph are connected is crucial for many graph algorithms. The graph might be stored in a specialized data structure if multiple queries are given. If the graph is static, using BFS and labeling the vertices gives an optimal data structure. However if edge insertions or deletions are allowed things get more complicated. If both insertions and deletions are allowed, the data structure of Holm et al. [4] which can handle edge updates in time $O(n \log^2 n)$ and queries in amortized time $O(\frac{\log n}{\log \log n})$ is the best currently known data structure for general graphs.

We consider the problem of constructing a deletion-only dynamic connectivity data structure for disk graphs. Given a set $S \subseteq \mathbb{R}^2$ of n sites with associated radii r_s, the disk graph is the intersection graph of the disks D_s induced by the sites and their radii. While the description of $D(S)$ has size $O(n)$, it might have $\Theta(n^2)$ edges. So when starting with some disk graph and subsequently deleting sites, over time up to $\Theta(n^2)$ edges are deleted. We describe a data structure whose overall running time for the sequence of n site deletion is $o(n^2)$. For unit disk graphs, such a data structure is already known [6].

On a high level, our approach works as follows. We define a sparse proxy graph $H = (V, E)$ with $S \subseteq V$ which perfectly represents the connectivity in $D(S)$. We then store H in the data structure of Holm et al. and describe how to update H on the deletion of a site.

2 Preliminaries

Our data structures relies on combining various data structures and techniques. We briefly recall their definition and relevant properties here.

* Supported in part by grant 1367/2016 from the German-Israeli Science Foundation (GIF).
† Supported by the German Science Foundation within the research training group 'Facets of Complexity' (GRK 2434).
‡ Supported in part by ERC StG 757609.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Without loss of generality we scale and translate S such that the minimum distance between two sites is $1+\varepsilon$ and that the point set fits into a square with diameter $2^{\lceil \log \operatorname{diam}(S) \rceil}$. Furthermore we set all radii to $\min\{r_s, \operatorname{diam}(S)\}$. The *quadtree* on S is now defined in the usual fashion as a tree of degree at most 4 with the square with diameter $2^{\lceil \log \operatorname{diam}(S) \rceil}$ being the root. We will not explicitly distinguish between the cells and the vertices of the quadtree. Each cell of diameter 2^i which contains at least two sites is subdivided into up to four non-empty cells of diameter 2^{i-1}. By the assumptions on our point set, no cell of diameter 1 is subdivided further. As the height of the quadtree defined this way is $O(\log \operatorname{diam}(S))$, and thus does not depend on n, we consider the *compressed quadtree*. Let $\sigma_1, \ldots, \sigma_k$ be maximal path of vertices with degree one in the quadtree. In the compressed quadtree $\mathcal{Q}$ this path is replaced by the edge $\sigma_1\sigma_k$. It is well known that such a compressed quadtree has $O(n)$ vertices, height $O(n)$ and can be constructed in $O(n \log n)$ time [1, 3]. Given a cell $\sigma \in \mathcal{Q}$ we denote by $|\sigma|$ its diameter.

We will use a heavy path decomposition on $\mathcal{Q}$. Slightly adapting the terminology of Sleator and Tarjan [7], we call an edge uv of a tree *heavy*, if v is the first child of u that maximized the total number of nodes in the subtree rooted at v and *light* otherwise. A *heavy path* is a maximal path which consists of heavy edges. The set of all heavy paths is called the *heavy path decomposition*.

▶ **Lemma 2.1** (Sleator and Tarjan [7]). *Let T be a tree with n vertices. Then, the following properties hold:* ***1.*** *Every leaf-root path in T contains $O(\log n)$ light edges;* ***2.*** *every vertex of T lies on exactly one heavy path; and* ***3.*** *the heavy path decomposition of T can be constructed in $O(n)$ time.*

Let X be a linearly ordered set. We aim to find a set $\mathcal{X}$ of subsets of X, such that $|\mathcal{X}| = O(|X|)$ and every contiguous subsequence can be partitioned into $O(\log |X|)$ subsets from $\mathcal{X}$. Using a standard approach [2, 8] this can be achieved by building a perfect binary search tree with the elements of X in the leaves. The interval can now be represented by $O(\log n)$ subtrees along the search paths for its boundaries. Finally we will use the following data structure:

▶ **Lemma 2.2** (Reveal data structure (RDS), Kaplan et al. [5]). *Let R and B be site sets with $|R| + |B| = n$. There is a data structure which after preprocessing allows to delete sites from R and B. After deleting a site from R it reports all disks from B now disconnected from $\bigcup_{s \in R} D_s$. Preprocessing, deleting m sites from R and an arbitrary number of sites from B takes $O((n \log^5 n + m \log^9 n)\lambda_6(\log n))$ expected time and $O(n \log^3 n)$ expected space, where $\lambda_s(n)$ is the maximum length of a Davenport-Schinzel sequence.*

3 The Proxy Graph

In order to describe H we first augment the compressed quadtree $\mathcal{Q}$ on S by adding additional cells. Given a site $t \in S$, let σ be the cell with $t \in \sigma$ and $|\sigma| \leq r_t \leq 2|\sigma|$. Let $N(t)$ be the 13×13 neighborhood of σ. Observe that all sites s with $r_s \leq r_t$ which can form an edge with r_t lie in a disk of radius at most $2r_t \leq 4|\sigma|$. All cells intersecting or containing that disk are part of $N(t)$, see Figure 1. We augment $\mathcal{Q}$ by adding the neighborhood $N(t)$ of all sites and call the resulting tree $\mathcal{Q}'$.

Considering the heavy path decomposition $\mathcal{R}$ of $\mathcal{Q}'$, we further decompose each heavy path $R \in \mathcal{R}$ into *canonical paths* using the technique from section 2. The set $\mathcal{P}$ of all canonical paths received this way has the following properties, which follow from those of $\mathcal{R}$ and $\mathcal{P}$. See Figure 2 for an illustration of the lemma.

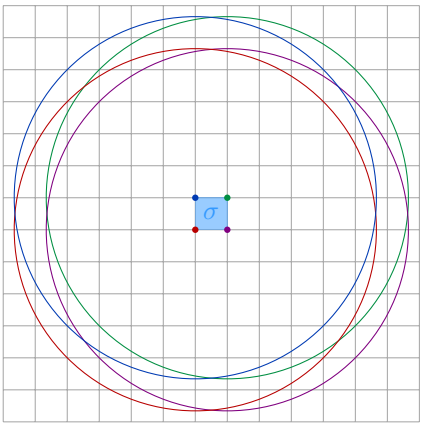


Figure 1 The disks with center in σ and radius $4|\sigma|$ are contained in $N_{13\times 13}(\sigma)$.

▶ **Lemma 3.1.** *Let σ be a cell of $\mathcal{Q}'$ and let π be the path in $\mathcal{Q}'$ from σ to the root. There exists a set $\mathcal{P}_\pi$ of canonical paths such that the following holds: **1.** $|\mathcal{P}_\pi| = O(\log^2 n)$, **2.** π is the disjoint union of the canonical paths in $\mathcal{P}_\pi$*

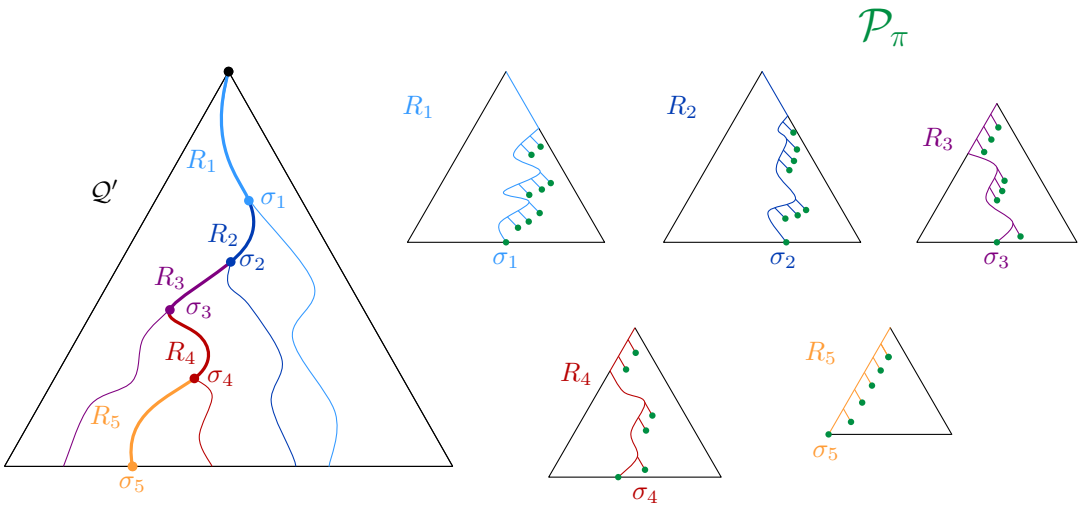


Figure 2 Illustration of Lemma 3.1. Left we see the decomposition of π into $O(\log n)$ heavy paths $R_1, \ldots R_k$. On the right the vertices defining $\mathcal{P}_\pi$ are depicted in green.

Given a constant $d \in \mathbb{N}$, let $\mathcal{C}_d$ be a set of d cones with common apex in the origin, each with opening angle $\frac{2\pi}{d}$, covering the plane. Let $P \in \mathcal{P}$ be a canonical path with smallest cell σ and largest cell τ. Then we denote the copy of the cones shifted to the center $a(\sigma)$ of σ by $\mathcal{C}_d(P)$. For constants d_1 and d_2 to be fixed later, now define $d_1 + d_2 + 1$ *regions* for P. These regions are partitioned into *outer regions*, *middle regions* and the *inner region*. The outer regions are obtained by considering $\mathcal{C}_{d_1}(P)$ and taking the intersection of each cone with the annulus with center $a(\sigma)$, outer radius $\frac{5}{2}|\sigma| + 2|\tau|$ and inner radius $\frac{5}{2}|\sigma|$. The middle regions are defined in a similar way, by taking the intersection of the cones in $\mathcal{C}_{d_2}(P)$ with the annulus with center $a(\sigma)$, outer radius $\frac{5}{2}|\sigma|$ and inner radius $|\sigma|$. Finally the inner region is defined as the disk with center $a(\sigma)$ and radius $|\sigma|$, see Figure 3 for an illustration. We call the set of the regions defined this way for all canonical paths $\mathcal{A}$.

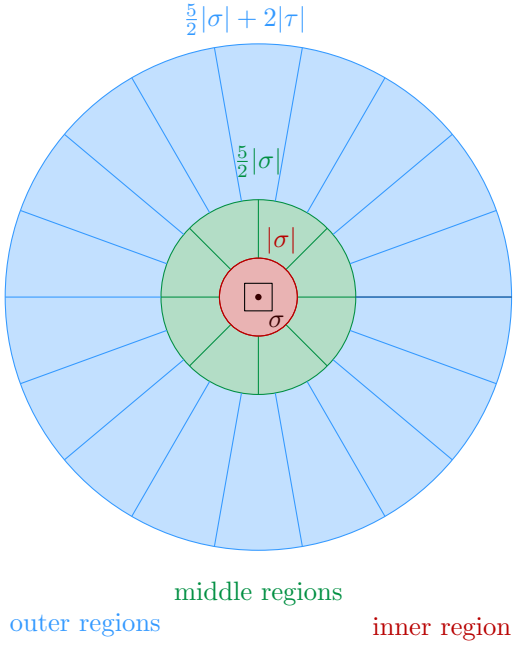


Figure 3 The regions defined by the smallest cell σ of a path

Based on these regions we define a sparse proxy graph H, which will be used to represent the connectivity in $D(S)$. The graph H has the vertex set $V = S \cup \mathcal{A}$. We will not strictly distinguish between the vertices of H and their associated sites and regions. With each region A we assign two sets $S_1(A) \subseteq S$ and $S_2(A) \subseteq S$. The set $S_1(A)$ will contain all sites contained in A whose are comparable to the size of the cells in the canonical path associated with A. The choice of the inner and outer radii in the definition of the regions together with our definition of comparable ensure that the disk graph induced by $S_1(A)$ is a clique. The set $S_2(A)$ contains sites with small radius which are contained in the smallest cell of the canonical path associated with A. These sites are chosen is such a way, that all edges in $D(S)$ between a site $s \in S_2(A)$ and $S_1(A)$ are represented with a single edge in H. As the sites in $S_1(A)$ form a clique this will not add a connection between sites in H which is not present in $D(S)$. Note that a site s will be contained in multiple sets $S_1(A)$ and $S_2(A)$ as it can lie in multiple regions and cells. Below we will give the details on the definition of $S_1(A)$ and $S_2(A)$. In Lemma 3.2 we show that the sites in $S_1(A)$ indeed form a clique. Then, in Lemma 3.3 we show that H accurately represents the connectivity of $D(S)$ and finally, as part of Lemma 4.1, we show that H is indeed sparse.

Now we give the details. The edge set of H is divided into two sets E_1 and E_2. The set E_1 is defined based on the sites in $S_1(A)$, whereas the set E_2 is defined based on $S_1(A)$. Let A be a region, where σ and τ are the smallest and largest cells respectively in the corresponding canonical path. If A is an outer region, let $S_1(A)$ be the set of all sites $t \in A$ with $|\sigma| \leq r_t \leq 2|\tau|$ and $\|a(\sigma)t\| \leq r_t + \frac{5}{2}|\sigma|$. On the other hand, if A is a middle or inner region, $S_1(A)$ consists of all sites $t \in A$ with $|\sigma| \leq r_t \leq 2|\tau|$. The edge set E_1 now consists of edges between A and all sites in $S_1(A)$.

Additionally we define a set $S_2(A)$ for the region. If A is an inner region, the set $S_2(A)$ consists of all sites s in σ with $r_s < |\sigma|$ which intersect at least one site in $S_1(A)$. In the other case, the set $S_2(A)$ contains all sites s in σ with $r_s \leq 2|\sigma|$, which are again intersecting at least one site in $S_1(A)$. The set E_2 now consists of edges between A and the sites in $S_2(A)$.

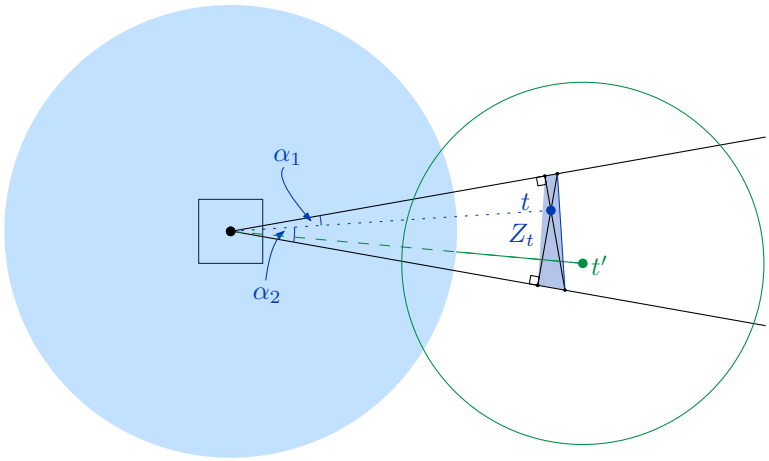


Figure 4 The part of the line segment $a(\sigma)t'$ in t' intersects the boundary of Z_t.

In order to show that this graph accurately represents the connectivity of the underlying disk graph $D(S)$ we first show that all sites within the same set $S_1(A)$ form a clique.

▶ **Lemma 3.2.** *Suppose that $d_1 \geq 18$ and $d_2 \geq 8$. Then, for any region $A \in \mathcal{A}$, the associated sites in $S_1(A)$ form a clique in $D(S)$.*

Proof (Sketch). As before, let σ be the smallest cell on the canonical path associates with A. If A is an inner or a middle regions this follows from considering the diameter of the regions. In both cases the diameter is at most $2|\sigma|$. As the sites in $S_1(A)$ have radius at least $|\sigma|$, the claim follows.

The case where A is an outer region is a bit more complicated. For each site $t \in S_1(A)$ we consider the two line segments going through t which are perpendicular to the lines defining the cone of A, and call the convex hull of these line segments Z_t, see Figure 4. Basic trigonometry shows that $Z_t \subseteq D_t$. Let $t' \in S_1(A)$ be a site with larger distance to $a(\sigma)$ than t. It can be argued that $t' \in S_1(A)$ either lies in Z_t or the part of the line segment $a(\sigma)t'$ which lies in D'_t intersects Z_t. In both cases it follows that the edge $\{t, t'\}$ exists in $D(S)$. ◀

Using Lemma 3.2 we can now show that H represents the connectivity in $D(S)$.

▶ **Lemma 3.3.** *Two sites $s, t \in S$ are connected in H if and only if they are connected in $D(S)$.*

Proof. First, we show that if s and t are connected in H they are also connected in $D(S)$. The graph H is bipartite, thus it suffices to show that if two sites u, u' are connected to the same region $A \in \mathcal{A}$, they are also connected in $D(S)$. This follows directly from Lemma 3.2: if u and u' are both in $S_1(A)$ they are part of the same clique and thus adjacent. If on the other hand $u \in S_2(A)$ or $u' \in S_2(A)$, the definition of $S_2(A)$ implies that $S_1(A)$, and the corresponding clique, is not empty. Thus there is a path from u through $S_1(A)$ to u'.

Now we consider two sites connected in $D(S)$ and show that they are also connected in H. If suffices to show that if s and t are connected by an edge in $D(S)$, they are connected to the same region $A \in \mathcal{A}$. Assume without loss of generality that $r_s \leq r_t$. Refer to Figure 5 for a depiction of the following argument. By the observation made above, there is a cell ρ in $N(t)$ which contains s. Consider the path π in $\mathcal{Q}'$ from the root to the smallest cell γ such that $s \in \gamma$ and $r_s \leq 2|\gamma|$. Then ρ lies on this path π. Let $\mathcal{P}_\pi$ be the decomposition of π

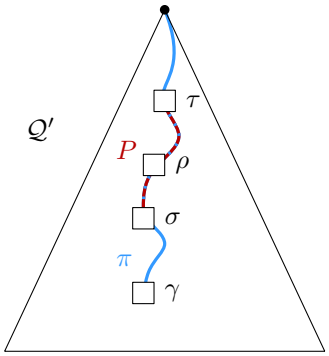


Figure 5 The cell γ is the smallest cell such that $r_s \leq 2|\gamma|$. The canonical path P contains ρ.

into canonical paths as defined in Lemma 3.1 and let P be the path containing ρ. Again let σ and τ be the smallest and largest cell on P respectively. By the definition of P we have $\gamma \subseteq \sigma \subseteq \rho \subseteq \tau$. As $\{s,t\}$ is an edge in $D(S)$, we have $\|st\| \leq r_s + r_t \leq 2|\sigma| + 2|\tau|$ and thus $\|a(\sigma)t\| \leq \frac{5}{2}|\sigma| + 2|\tau|$. This implies that t lies in a region A defined by P. If A is an inner region and $|\sigma| \leq r_s \leq 2|\sigma|$ then s is also part of $S_1(A)$ by definition, in the other case s is contained in $S_2(A)$. In both cases it follows that s and t are both connected to A. ◀

4 The Data Structure

The main idea of the data structure is to store H in a Holm et al. data structure $\mathcal{H}$ and use $\mathcal{H}$ to answer queries. The main challenge is to maintain H and $\mathcal{H}$ under deletion of sites, as well as efficiently preprocess the sites. The role of the components is shown in Figure 6.

▶ **Lemma 4.1.** *The graph H has $O(n)$ vertices, and $O(n \log^2 n)$ edges. The graph and a Holm et al. data structure $\mathcal{H}$ containing it can be constructed in $O(n \log^4 n)$ time. Within the same time bound we can construct the extended quadtree $\mathcal{Q}'$, the heavy path decomposition $\mathcal{R}$ and the binary search trees on the heavy paths.*

Proof (Sketch). The augmented compressed quadtree has $O(n)$ vertices. By adding appropriate *virtual sites* to the centers of suitable cells, it can be constructed in $O(n \log n)$ time. Constructing the heavy path decomposition on this tree takes time $O(n)$ by Lemma 2.1. The binary search trees defining the canonical paths also have a total $O(n)$ vertices and can be constructed in $O(n \log^2 n)$ overall time. As we define $O(1)$ regions for each canonical path, the bound on the number of vertices follows.

The number of edges follows from the total size of the sets $S_1(A)$ and $S_2(A)$. For the total size of the sets $S_1(A)$ consider a site t and its neighborhood $N(t)$. Each cell in $N(t)$ is contained in $O(\log n)$ canonical paths. In order to satisfy both the distance and the radius constraint for the set $S_1(A)$ of a region, its associated path has to contain a cell of $N(t)$ and thus t is contained in $O(\log n)$ sets $S_1(A)$.

A site s is contained in the sets $S_2(A)$ along the canonical paths which decompose the path π from the root to the smallest cell γ in $\mathcal{Q}'$ with $s \in \gamma$ and $r_s \leq 2|\gamma|$, again refer to Figure 5. By Lemma 3.1 there are $O(\log^2 n)$ such regions. Summing up over all sets $S_1(A)$ and $S_2(A)$ this results in $O(n \log^2 n)$ overall edges.

The sets $S_1(A)$ can be found by following the argument about the size and explicitly checking the conditions for each canonical path containing a given cell of $N(t)$. In order to find the sets $S_2(A)$ we first have to construct an additively weighted voronoi diagram on each

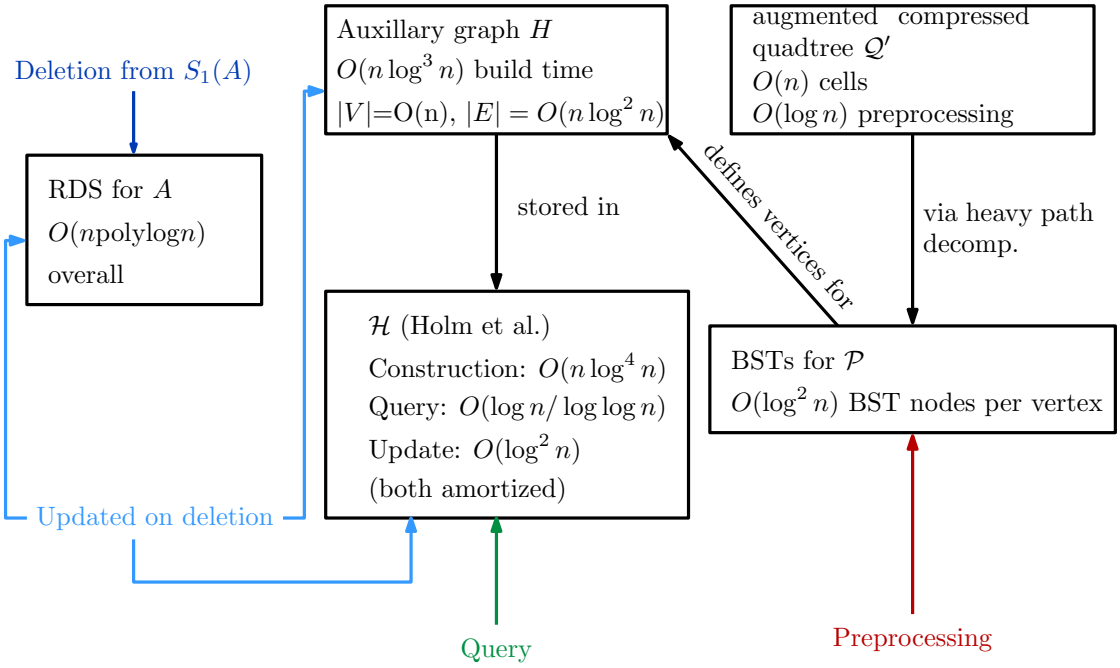


Figure 6 Components of the data structure and their connections

set $S_1(A)$. Assigning each site $t \in S_1(A)$ the weight $-r_t$, allows us to determine in $O(\log n)$ time if a site s intersects some site in $S_1(A)$. Performing this query for all regions along the path defined in the size argument yields all sets $S_2(A)$. Combining the steps for finding $S_1(A)$ and $S_2(A)$ finds all edges in $O(n \log^3 n)$ time. Inserting the edges one by one to the Holm et al. data structure $\mathcal{H}$ dominates the time, leading to $O(n \log^4 n)$ overall time. ◀

We simply perform connectivity queries on $\mathcal{H}$. To handle deletions, we build one reveal data structure (RDS) for each region $A \in \mathcal{A}$. We set $R' = S_1(A)$ and $B' = S_2(A)$ for each RDS. Let n' be the total number of sites stored in one of the data structures, then preprocessing, deleting some sites from B', deleting m' sites from R' and retrieving the revealed sites takes $O((n' \log^5 n + m' \log^9 n)\lambda_6(\log n))$ total time by Lemma 2.2.

When deleting a site s, we can safely delete it from all sets $S_2(A)$ containing it, together with the associated edges in E_2 and the sites stored in the RDS of A. When deleting a site from a set $S_1(A)$ however it is possible that some other sites have to be removed from the associated set $S_2(A)$. These sites are reported by the RDS on deleting s from R', allowing us to safely delete them from $S_2(A)$. Summing up we get the following result.

▶ **Theorem 4.2.** *The data structure described above answers connectivity queries in amortized time* $O\left(\frac{\log n}{\log \log n}\right)$ *with* $O((n \log^7 n + m \log^{11} n)\lambda_6(\log n))$ *overall expected update time for* m *deletions.*

References

1 Kevin Buchin, Maarten Löffler, Pat Morin, and Wolfgang Mulzer. Preprocessing imprecise points for Delaunay triangulation: Simplified and extended. *Algorithmica*, 61(3):674–693, 2011. doi:10.1007/s00453-010-9430-0.

2 Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and applications*. Springer-Verlag, Berlin, 3rd edition, 2008. doi:10.1007/978-3-540-77974-2.

3 Sariel Har-Peled. Quadtrees–hierarchical grids. In *Geometric Approximation Algorithms*, volume 173 of *Mathematical Surveys and Monographs*, chapter 2. American Mathematical Society, 2011. doi:10.1090/surv/173.

4 Jacob Holm, Kristian de Lichtenberg, and Mikkel Thorup. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. *J. ACM*, 48(4):723–760, 2001. doi:10.1145/502090.502095.

5 Haim Kaplan, Alexander Kauer, Wolfgang Mulzer, and Liam Roditty. Sampling hyperplanes and revealing disks. Abstract submitted to EuroCG21., 2021.

6 Haim Kaplan, Wolfgang Mulzer, Liam Roditty, and Paul Seiferth. Dynamic connectivity for unit disk graphs. In *Proc. 32nd European Workshop on Computational Geometry (EWCG)*, 2016.

7 Daniel D. Sleator and Robert Endre Tarjan. A data structure for dynamic trees. *J. Comput. Syst. Sci.*, 26(3):362–391, 1983. doi:10.1016/0022-0000(83)90006-5.

8 Dan E. Willard and George S. Lueker. Adding range restriction capability to dynamic data structures. *J. ACM*, 32(3):597–617, 1985. doi:10.1145/3828.3839.

Maximum-Width Rainbow-Bisecting Empty Annulus

Sandip Banerjee[1], Arpita Baral [*2], and Priya Ranjan Sinha Mahapatra [†2]

1 **The Hebrew University of Jerusalem, Jerusalem, Israel.**
sandip@cs.huji.ac.il

2 **Department of Computer Science and Engineering, University of Kalyani, kalyani, India.**
arpitabaral@gmail.com, priya@klyuniv.ac.in

Abstract

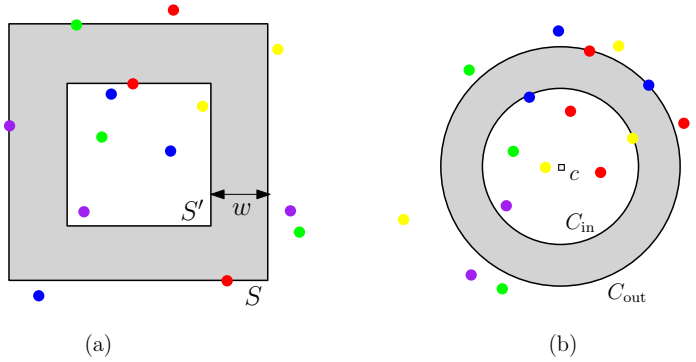
Given a set of n colored points with k colors in the plane, we study the problem of computing a maximum-width rainbow-bisecting empty annulus (of objects specifically axis-parallel square and circle) problem. We call a region *rainbow* if it contains at least one point of each color. The maximum-width rainbow-bisecting empty annulus problem asks to find an annulus A of a particular shape with maximum possible width such that A does not contain any input points and it bisects the input point set into two parts, each of which is a *rainbow*. We compute a maximum-width rainbow-bisecting empty axis-parallel square and circular annulus in $O(n^3 + n^2 k \log k)$ time using $O(n \log n)$ space and in $O(n^3)$ time using $O(n^2)$ space respectively.

1 Introduction

In the context of facility location most of the existing literature deals with the placement of the facilities (e.g. pipelines) among a set of customers (represented by points in $\mathbb{R}^2$) but there are scenarios where the facilities are hazardous (e.g pipelines transporting toxic materials). In these scenarios the objective is maximizing the minimal distance between the hazardous facility and the given customers. Considering this situation, some problems have been studied in literature that computes a widest L-shaped empty corridor [8], a widest 1-corner corridor [11], a largest empty annulus [12]. For empty annulus we additionally impose another constraint that each of the two non-empty regions corresponds to two *self-sustained* smart cities, meaning that each of the smart cities is comprised of essential facilities of each type such as schools, hospitals, etc. This situation calls for the computation of two rainbow regions separated by an empty region where empty region contains hazardous facility and each color represents each essential facilities in each of the rainbow region. Given a set of n points in $\mathbb{R}^2$, each point colored with one of the k ($k \leq n$) colors, a *rainbow* (or color spanning) region in $\mathbb{R}^2$ contains at least one point of each color. Abellenas et al. first proposed algorithms for computing a smallest color-spanning axis-parallel rectangle [2], narrowest strip [2], circle [1] in $O(n(n-k)\log^2 k)$, $O(n^2\alpha(k)\log k)$ and $O(kn \log n)$ respectively. Later the time complexities to compute a smallest color-spanning rectangle and a narrowest color-spanning strip were improved to $O(n(n-k)\log k)$ and $O(n^2 \log n)$ by Das et al. [9]. Kanteimouri et al. [16] computed a smallest axis-parallel color-spanning square in $O(n \log^2 n)$ time. Hasheminejad et.al [15] proposed an $O(n \log n)$ time algorithm to compute a smallest color-spanning

* The author would like to thank Stefan Langerman for his insightful discussions and comments.
† Research supported in part by SERB, Govt. of India, Ref. No. MTR/2019/000792.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.



(a) (b)

Figure 1 (a) A RBSA of width w with outer and inner squares S and S'. (b)A RBCA with outer & inner disk C_{out} & C_{in} & common center c.

equilateral triangle. The shortest color-spanning interval, smallest color-spanning t squares and smallest color-spanning t circles have been studied by Banerjee et al. [17] and they have given some hardness and tractability results from parameterized complexity point of view. Acharyya et al. [3] computed a minimum-width color-spanning annulus for circle, equilateral triangle, axis-parallel square and rectangle in $O(n^3 \log n)$, $O(n^2 k)$, $O(n^3 + n^2 k \log k)$ and $O(n^4)$- time respectively. D. Báñez et al. [12] first studied the largest empty circular annulus problem and improved to $O(n^3)$-time [13]. Bae et al. [7] proposed algorithms computing a maximum-width empty axis-parallel square and rectangular annulus in $O(n^3)$ and $O(n^2 \log n)$ time respectively. Recently Erkin et al. [5] studied a variant of a covering problem where the objective is to cover a set of points by *conflict free* set of objects, where an object is *conflict free* if it covers at most one point from each color class. We compute here a *maximum-width rainbow-bisecting empty axis-parallel square annulus* in $O(n^3 + n^2 k \log k)$ time using $O(n \log n)$ space. Notice on computing this, a subcase of our objective maps to the problem of computing a largest *rainbow-bisecting empty axis-parallel L corridor* and solve it in $O(n \log^4 n)$-time. This improves the time complexity of the algorithm for computing a *widest empty axis-parallel L corridor* from $O(n^2 \log n)$ [7] to $O(n \log^3 n)$ time. We also propose an $O(n^3)$ time algorithm for computing a *maximum-width rainbow-bisecting empty circular annulus* in $O(n^2)$ space.

2 Definition and Terminologies

We are given a set P containing n points in $\mathbb{R}^2$ where each point is colored with one of the given colors $\{1, \ldots, k\}$ and $k \leq n$. For each color $i \in [k]$ there exists at least two points $a, b \in P$ of color i. For any point $p \in P$, we denote its x- and y-coordinates, color by $x(p)$, $y(p)$ and $\alpha(p)$ respectively. We borrow the definition of center, radius of an axis-parallel square from Bae et al. [7]. For an axis-parallel square S, the *(inward) offset* of S by a real number $\delta \geq 0$ is a smaller axis-parallel square obtained by sliding each side (top, bottom, left and right) of S inwards by δ. An *axis-parallel square annulus* (Ref. Fig. 1(a)) is the region between an axis-parallel square S and its offset S' by some $\delta \geq 0$, where S, S', and δ are the *outer* square, *inner* square, and the *width* of the annulus respectively. We allow the outer and inner squares of an axis-parallel square annulus to have one or more sides *at infinity* (∞) which means the associated coordinate value of that side is infinite. A *circular annulus* is the region between two concentric disks C_{out} (outer disk) and C_{in} (inner disk) where $C_{\text{in}} \subseteq C_{\text{out}}$, center c, radius of outer disk $ra_{C_{\text{out}}}$ and radius of inner disk $ra_{C_{\text{in}}}$ and $ra_{C_{\text{in}}} \leq ra_{C_{\text{out}}}$. The

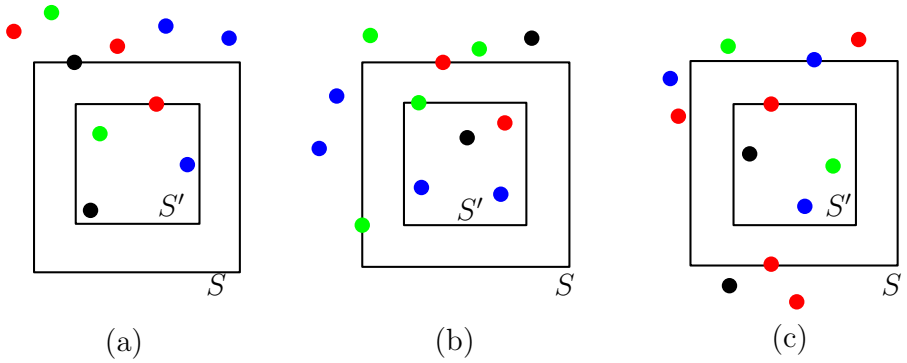


Figure 2 (a) $C1$ Configuration (b) $C2$ Configuration (c) $C3$ Configuration

width is defined as the difference of the radii, $ra_{C_{out}} - ra_{C_{in}}$ (Ref. Fig.1(b)). An axis-parallel square annulus (resp. a circular annulus) A is said to be *rainbow-bisecting empty* if it does not contain any point of P in its interior and divides P into two non-empty subsets such that each subset is a rainbow, one subset lies within or on the boundary of the inner square (resp. inner disk) of A and the other subset lies outside or on the boundary of the outer square (resp. outer disk) of A. From now we refer an axis-parallel square (resp. axis-parallel square annulus) as square (resp. square annulus). Any *rainbow-bisecting empty square annulus* and *rainbow-bisecting empty circular annulus* are denoted by RBSA and RBCA respectively.

3 Maximum-Width Rainbow-Bisecting Empty Square Annulus

In this section, we compute a maximum-width RBSA from a given point set P on $\mathbb{R}^2$.

▶ **Lemma 3.1.** *A maximum-width RBSA A must contain at least one point of P on one side of both S (outer square) and S' (inner square) and all the potential configurations of A can be mapped in the following three configurations (Ref. Fig. 2).*

- *$C1$: At least one side of both S, S' contains a point $\in P$, remaining sides of S are at ∞.*
- *$C2$: One side of S' contains a point of P and any two adjacent sides of S are defined by points $\in P$ and the other two sides of S are at ∞.*
- *$C3$: One side of S' and one pair of opposite sides of S must contain points $\in P$.*

We propose different algorithms for handling above three different configurations and report the maximum. As a part of preprocessing, we sort the points $\in P$ w.r.t both the co-ordinates.

$C1$ configuration: A RBSA of configuration $C1$ maps to an empty strip such that regions on both side of the strip are rainbow (called as *rainbow-bisecting empty strip*, RBES).

▶ **Lemma 3.2.** *A maximum-width RBES can be computed in $O(n)$ time using $O(n)$ space.*

$C2$ configuration: Observe solving this configuration corresponds to the problem of finding a largest *rainbow-bisecting* empty axis-parallel L-shaped corridor (RBLC) (w.l.o.g. we assume pointing down and right; Ref. Fig. 3(a)). Our algorithm maintains a horizontal sweepline H (moving from bottom to top) which on encountering a point $p_i \in P$ calls the following decision problem, DP as the main subroutine and it does for all the points $\in P$.

Given: A positive real $w > 0$ *Task*: Does there exist a RBLC with horizontal part right above H and of width w ?

The DP eventually computes a maximum-width RBLC from the set of possible widths $w_{p_{i'}p_{j'}}$ where $w_{p_{i'}p_{j'}} \in \{[y(p_{i'}) - h, y(p_{j'}) - h]\}$, $i' < j' \leq n$ & $p_{i'}$, $p_{j'}$ are any 2 consecutive

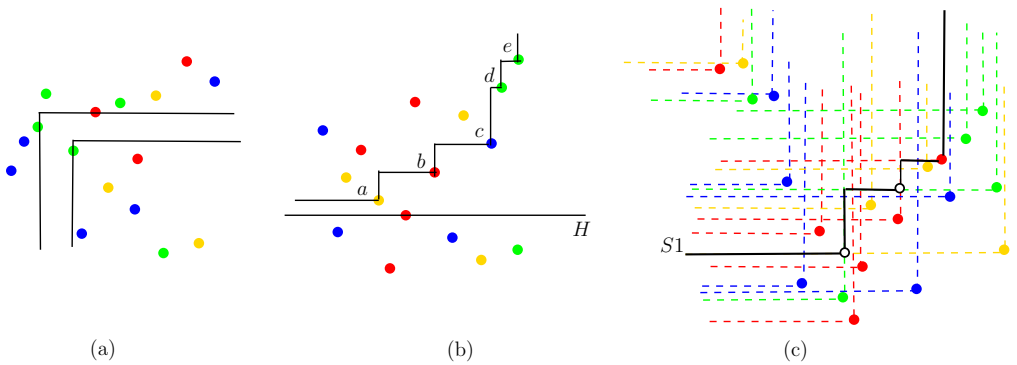


Figure 3 (a) A rainbow-bisecting empty L corridor. (b) $a, \ldots, e$ are the dominating points above H. (c) Staircase $S1$ for point set P, the empty circles denote staircase points $\notin P$.

points on the *staircase* above $y(H) = h$ (A point p_i *dominates* p_j if both (i)$x(p_j) > x(p_i)$ & $(ii)y(p_j) > y(p_i)$ hold; p_i, p_j are points on the *staircase* where any two points p_i, p_j lying on the staircase satisfies $x(p_i) < x(p_j) \implies y(p_i) < y(p_j)$); Ref. Fig. 3(b)).

We maintain the following data structures for answering queries to be used by the DP.

- Store all the points $\in P$ in a 2D range tree [10] $T1$ in order to find the *dominating* points above H. The first dominating point say, p_a, above $y(H) = h$ is the point with the smallest y-coordinate above H. We find the next dominating point, say p_b by querying $\arg\min_j \{y(p_j) | x(p_j) > x(p_a)$ && $y(p_j) > y(p_a)\}$ on $T1$. Similarly find the rest.
- Populate a 1D range tree [10] $T2$ with the x-coordinate values of the points lying below H. Also each node $v \in T2$ stores the maximum *horizontal gap* (difference in x-coordinate values of 2 consecutive sorted (w.r.t. x values) points) in the canonical subset (CS) of v. The maximum horizontal gap for a given range, say $[x_0, +\infty]$ can be found by comparing $O(\log n)$ values including every gap between 2 consecutive CS.
- Consider two *staircases* $S1$ and $S2$, such that any empty L-shaped corridor lying in the region above $S1$ and below $S2$ is a RBLC. The staircase $S1$ is generated as follows: We traverse the points $\in P$ from bottom to top (say initially we are at $x = -\infty$) and maintain a counter on the number of points of each color we traversed so far. Next in order to get the leftmost point of $S1$ we move towards the right until the counter of a color (say for instance color $j \in k$) becomes zero. Next from the leftmost point of $S1$ we move upwards until the count for color j becomes 1. We repeat the above procedure in order to determine the points of $S1$ until all the points are being traversed (Ref. Fig. 3(c)). In a similar way we construct staircase $S2$ by traversing the points of P from top to bottom. It can be easily verified from the above construction that any empty L-shaped corridor lying below $S1$ as well as above $S2$ cannot be a RBLC.

  Construct two balanced BST $T(S1)$ and $T(S2)$ to store the points of two *staircases* $S1$ and $S2$ respectively. Insert the points $\in P$ on $T(S1)$ (resp. on $T(S2)$) during traversing from bottom to top (resp. top to bottom) according to their x-coordinate values. The points of $S1$ and $S2$ are marked on $T(S1)$ and $T(S2)$ respectively.

  We verify a w-width empty axis-parallel L-shaped corridor is a RBLC or not by checking if the top-left corner points (inside and outside) lies above $S1$ and below $S2$ or not. For *inside* top-left corner point (c_i), find immediate left $(a1)$ and right $(a2)$ points of $x(c_i)$ that lies on $S1$ from $T(S1)$ and see if $y(c_i) > y(a_2)$ or not. Similarly

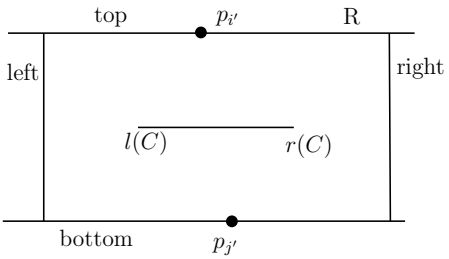


Figure 4 Centers of candidate outer squares in R lie in the segment $[l(C), r(C)]$, $l(C) = (max(x(p_{i'}), x(p_{j'})) - r, y(l))$ and $r(C) = min(x(p_{i'}), x(p_{j'})) + r, y(l))$.

verify for *outside* top-left corner point from $S2$.

The above discussion leads to the following result.

▶ **Theorem 3.3.** *Given n points in $\mathbb{R}^2$ where each one is colored with one of the k colors, a maximum-width RBLC can be found in $O(n \log^4 n)$ time using $O(n \log n)$ space.*

Here we mention that our algorithm improves the time complexity for computing a *widest empty axis parallel L corridor* from $O(n^2 \log n)$ (Theorem 1 of [7]) to $O(n \log^3 n)$.

$C3$ configuration: We start this case with the following observation.

▶ Observation 1. The potential locations of center of the outer square, S for a fixed $p_{i'}$ and $p_{j'}$ lies on the line ℓ $(y(\ell) = (y(p_{i'}) + y(p_{j'}))/2)$, where $p_{i'}$ and $p_{j'}$ defines the top and bottom sides of S respectively (Ref. Fig. 4).

Note that the outer square, S of a potential candidate annulus for a fixed $p_{i'}$ and $p_{j'}$ lies inside a rectangle R (Ref. Fig. 4) where $x(\text{left}) = \max\{x(p_{i'}), x(p_{j'})\} - 2r$ and $x(\text{right}) = \min\{x(p_{i'}), x(p_{j'})\} + 2r$ $(r := (y(p_{i'}) - y(p_{j'}))/2)$.

For a fixed outer square $S(c)$ with center c lies on $C \subset \ell$ we compute the radius of the inner square $S'(c)$ by finding the farthest point from center c lying inside $S(c)$. We first plot L_∞ distance of each point $p \in P_{i'j'} \forall c \in C$ in R (denoted by $f_p(c)$, discussed in [6]), where $P_{i'j'}$ is the set of points inside R. Next consider the distance functions of all points of each color i followed by computing the upper envelope for each color i from $\cup_{\alpha(p)=i} f_p(c)$, denoted by Φ_i. We get the centers of potential RBSA by projecting the breakpoints of the lower envelopes of all Φ_i functions on segment C [3]. Finally we construct the inner squares of the corresponding centers by finding the farthest points from the breakpoints of the upper envelope $F(c) := \max_{p \in P_{i'j'}} f_p(c)$ [7]. Considering all choices of $p_{i'}$ and $p_{j'}$ and the above discussion leads to the following result.

▶ **Lemma 3.4.** *A potential maximum-width RBSA corresponding to $C3$ configuration can be computed in $O(n^3 + n^2 k \log k)$ time and $O(n)$ space.*

We conclude the section with the following theorem.

▶ **Theorem 3.5.** *Given n points in $\mathbb{R}^2$ where each one is colored with one of the k colors, a maximum-width RBSA can be computed in $O(n^3 + n^2 k \log k)$ time using $O(n \log n)$ space.*

4 Maximum-width Rainbow-Bisecting Empty Circular Annulus

In this section, we compute a maximum-width RBCA from a given point set P on $\mathbb{R}^2$. Here we assume no four input points are concyclic.

▶ **Lemma 4.1.** *A maximum-width RBCA A is defined by four points from P resulting one of the following potential configurations-(i) Type 1 (resp. Type 2) - C_{out} (resp. C_{in}) has three*

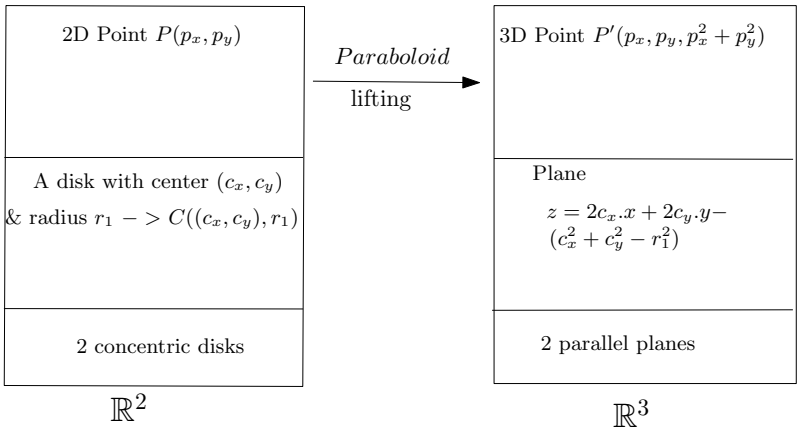


Figure 5 Paraboloid transformation of the input points. Left side maps right side.

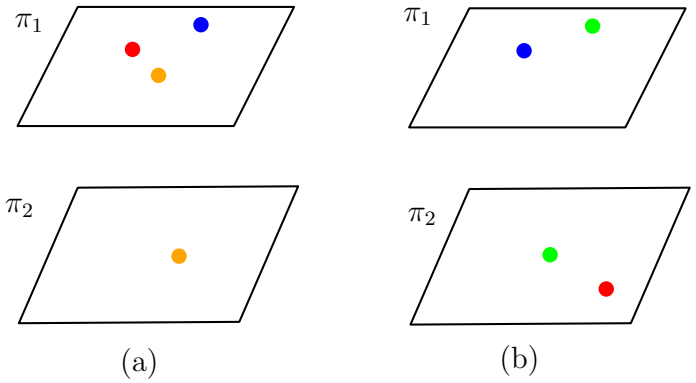


Figure 6 (a) A C_{31} empty slab defined by 2 parallel planes π_1 and π_2. (b) A C_{22} empty slab.

points of P and C_{in} (resp. C_{out}) contains one point of P and (ii) Type 3 - Each of C_{in} and C_{out} are defined by two points of P.

Adopting the technique of D.Báñez et al. [13], transform the points $\in P$ from $\mathbb{R}^2$ to $\mathbb{R}^3$ using paraboloid lifting (Ref. Fig. 5) resulting the problem of computing the largest empty circular annulus problem in $\mathbb{R}^2$ being mapped to the largest empty slab problem (LESP) in $\mathbb{R}^3$.

> Largest Empty Slab Problem: (LESP)
> *Given*: A set S of n points in $\mathbb{R}^3$
> *Task*: Find a *widest empty slab* where a *slab* through S is the open region of $\mathbb{R}^3$ that is bounded by 2 parallel planes intersecting the convex hull of S. The *width* of the slab is the distance between the bounding planes.

A candidate annulus (in $\mathbb{R}^2$) of Type 1 (or 2) & Type 3 is being mapped (in $\mathbb{R}^3$) to empty slabs of Type C_{31} and C_{22} respectively (Ref. Fig. 6). Finally, the optimal solution is obtained by tracking the slabs of Type C_{31} and C_{22} which span k colors on both sides of it. Generate candidate empty slabs of Types C_{31}, C_{22} using topological sweep [4, 14] on the arrangement $\mathcal{A}(H)$ of dual planes (each plane has a color) corresponding to the points in primal. During the sweep we maintain a count on the number of planes of each color $\in [k]$ present above and below for each edge of the current planar cut of each plane. Once the initial planar cut for plane $k' \in \mathcal{A}(H)$ is determined, we initialize arrays namely $UP_{k'}[1:k]$, $LOW_{k'}[1:k]$ and

$CO_{k'}[1 : n-1]$ following sequence of edges in $N_{k'}[1 : n-1]$, where $N_{k'}[1 : n-1]$ is a list of pairs of indices indicating the lines delimiting each edge of the current planar cut for plane k'. The array $CO_{k'}[i] = (a, b)$, indicates the number of colored planes above and below each edge of $N_{k'}[i]$ of current planar cut for k' where $UP_{k'}$ and $LOW_{k'}$ help to initialize $CO_{k'}$. We get each $CO_{k'}$ in $O(n)$ time. As the sweep performs an elementary step, the incoming edges are swapped at the corresponding vertex in each planar cut and $CO_{k'}$ is updated following new $N_{k'}$ in $O(1)$ time. The above discussion along with lemma 4 [13] leads to the following result.

▶ **Theorem 4.2.** *Given a set of n points in $\mathbb{R}^2$, each one is colored with one of the k colors, the maximum-width RBCA problem can be solved in $O(n^3)$ time and $O(n^2)$ space.*

We extend the above idea to compute a maximum-width $RBCA$ whose center lies on a given line L in the plane. We have the following lemma.

▶ **Lemma 4.3.** *A maximum-width RBCA A whose center lies on a given line L is defined by three points of P where C_{in} (resp. C_{out}) contains two points of P and C_{out} (resp. C_{in}) contains one point of P.*

Proof. Similar as described in lemma 4.1. ◀

We conclude this section with the following corollary.

▶ **Corollary 4.4.** *A maximum-width RBCA A whose center lies on a given query line L in $\mathbb{R}^2$ can be computed in $O(n^2)$ time and space.*

References

1. Manuel Abellanas, Ferran Hurtado, Christian Icking, Rolf Klein, Elmar Langetepe, Lihong Ma, Belén Palop, and Vera Sacristán. The farthest color Voronoi diagram and related problems. In *Proc. 17th EuroCG 2001*, pages 113–116, 2001.
2. Manuel Abellanas, Ferran Hurtado, Christian Icking, Rolf Klein, Elmar Langetepe, Lihong Ma, Belén Palop, and Vera Sacristán. Smallest color-spanning objects. In *Algorithms - ESA 2001, 9th Annual European Symposium, Aarhus, Denmark, August 28-31, 2001, Proceedings*, pages 278–289, 2001.
3. Ankush Acharyya, Subhas C. Nandy, and Sasanka Roy. Minimum width color spanning annulus. *Theor. Comput. Sci.*, 725:16–30, 2018.
4. Efthymios Anagnostou, Vassilios G. Polimenis, and Leonidas J. Guibas. Topological sweeping in three dimensions. In *Proceedings of the International Symposium on Algorithms, SIGAL*, volume 450 of *Lecture Notes in Computer Science*, pages 310–317, 1990.
5. E. M. Arkin, A. Banik, P. Carmi, G. Citovsky, M. J. katz, J. S. B. Mitchell, and M. Simakov. Selecting and covering colored points. *Discrete Applied Mathematics.*, 250:75–86, 2020.
6. Sang Won Bae. Computing a minimum-width square annulus in arbitrary orientation. *Theoret. Comput. Sci.*, 718:2–13, 2018.
7. Sang Won Bae, Arpita Baral, and Priya Ranjan Sinha Mahapatra. Maximum-width empty square and rectangular annulus. In *Proc. Algorithms and Computation - 13th International Conference, WALCOM 2019*, pages 69–81, 2019.
8. Siu-Wing Cheng. Widest empty L-shaped corridor. *Inform. Proc. Lett.*, 58(6):277 – 283, 1996.
9. Sandip Das, Partha P. Goswami, and Subhas C. Nandy. Smallest color-spanning object revisited. *Int. J. Comput. Geometry Appl.*, 19(5):457–478, 2009.
10. Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag TELOS, Santa Clara, CA, USA, 3rd ed. edition, 2008.

11 J.M. Díaz-Báñez, M.A. López, and J.A. Sellarès. On finding a widest empty 1-corner corridor. *Inform. Proc. Lett.*, 98(5):199 – 205, 2006.

12 José Miguel Díaz-Báñez, Ferran Hurtado, Henk Meijer, David Rappaport, and Joan Antoni Sellarès. The largest empty annulus problem. *Int. J. Comput. Geom. Appl.*, 13(4):317–325, 2003.

13 José Miguel Díaz-Báñez, Mario Alberto López, and Joan Antoni Sellarès. Locating an obnoxious plane. *Eur. J. Oper. Res.*, 173(2):556–564, 2006. URL: `https://doi.org/10.1016/j.ejor.2005.02.048`.

14 Herbert Edelsbrunner and Leonidas J. Guibas. Topologically sweeping an arrangement. *J. Comput. Syst. Sci.*, 38(1):165–194, 1989. URL: `https://doi.org/10.1016/0022-0000(89)90038-X`.

15 Javad Hasheminejad, Payam Khanteimouri, and Ali Mohades. Computing the smallest color-spanning equilateral triangle. In *Proc. 31st EuroCG 2015*, pages 32–35, 2015.

16 Payam Khanteimouri, Ali Mohades, Mohammad Ali Abam, and Mohammad Reza Kazemi. Computing the smallest color-spanning axis-parallel square. In *Algorithms and Computation - 24th International Symposium, ISAAC 2013, Proceedings*, pages 634–643, 2013.

17 S.Banerjee, N. Misra, and S. Nandy. Color spanning objects: Algorithms and hardness results. *Discrete Applied Mathematics.*, 280:14–22, 2020.

Axis-Aligned Square Contact Representations

Andrew Nathenson[*1]

1 **California State University Northridge**
andrew.nathenson.540@my.csun.edu

Abstract

We introduce a new class $\mathcal{G}$ of plane bipartite graphs and prove that each graph in $\mathcal{G}$ admits a proper square contact representation. A contact between two squares is *proper* if they intersect in a line segment of positive length. The class $\mathcal{G}$ is the family of quadrangulations obtained from the 4-cycle C_4 by successively inserting a single vertex or a 4-cycle of vertices into a face.

1 Introduction

Geometric representations of graphs have many applications and yield intriguing problems [9]. Koebe's celebrated *circle packing theorem* [8], for example, states that every planar graph is a contact graph of interior-disjoint disks in the plane. Shramm [11] proved that this theorem holds even if we replace the disks with homothets of an arbitrary smooth strictly convex body in the plane. The result extends to non-smooth convex bodies in a weaker form (where a homothet may degenerate to a point, and three or more homothets may have a common point of intersection), and every planar graph is only a *subgraph* of such a contact graph.

In this paper, we consider *strong* contact representations with interior-disjoint convex bodies where no three convex bodies have a point in common. It is an open problem to classify graphs that admit a strong contact representation with homothets of a triangle or a square. It is known that every partial 3-tree [1] and every 4-connected planar graph admits a strong contact representation with homothetic triangles, see [5, 6]; but there are 3-connected planar graphs which do not admit such a representation. We note here that every planar graph admits a strong contact representation with (non-homothetic) triangles [3]; see also [6].

Strong contact representations with homothetic squares have been considered only recently. Da Lozzo et al. [2] proved that every $K_{3,1,1,}$-free partial 2-tree admits a proper contact representation with homothetic squares, where a contact between two squares is *proper* if they intersect in a line segment of positive length (in particular, proper contacts yield a strong contact representation). Eppstein [4] indicated that another family of graphs, defined recursively, can also be represented as a proper contact graph of squares. We remark that Klawitter et al. [7] proved that every triangle-free planar graph is the proper contact graph of (non-homothetic) axis-aligned rectangles.

Contribution. Let $\mathcal{G}$ be a family of plane bipartite graphs defined recursively as follows. (i) $\mathcal{G}$ contains the 4-cycle C_4. (ii) If $G \in \mathcal{G}$ and $f = (v_1, v_2, v_3, v_4)$ is a bounded 4-face of G, then $\mathcal{G}$ also contains the graphs G_a and G_b obtained by the following two operations: (a) insert a vertex u into f and connect it to v_1 and v_3; (b) insert four vertices $u_1, \ldots, u_4$ into f, add the cycle (u_1, u_2, u_3, u_4) and the edges $u_i v_i$ for $i = 1, \ldots, 4$; see Fig. 1.

Every 2-degenerate bipartite plane graph can be constructed by operation (a); and the 1-skeleton of every polycube whose dual graph is a tree [4] can be constructed by operation

* Research on this paper was partially supported by the NSF award DMS-1800734.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

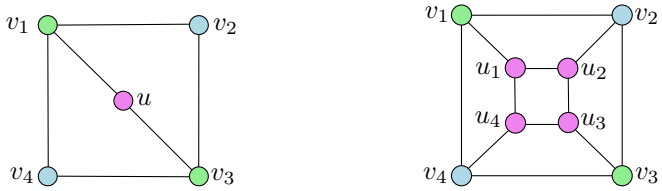


Figure 1 The two operations used to obtain a graph in $\mathcal{G}$

(b). However, the two operations jointly produce a larger class $\mathcal{G}$, which belongs to the class of 3-degenerate bipartite plane graphs. The following is main result of this paper.

▶ **Theorem 1.** *Every graph in $\mathcal{G}$ admits a proper square contact representation.*

Terminology. Let $G = (V, E)$ be an edge-maximal plane bipartite graph. In a square contact representation, every vertex v_i corresponds to an axis-aligned square $s(v_i)$, and every face to an axis-aligned rectangle $g(f_i)$, which is also called the *gap* corresponding to f_i. The aspect ratio of an axis-aligned rectangle r is height(r)/width(r). The side length of a square s is denoted by len(s). Scaling up a square from a corner by (or to) x means to increase the width and height of the square by x (or to x) in such a way that the position of the specified corner remains fixed.

2 Maintaining a Square Contact Representation

In this section, we show how to maintain a square contact representation of a graph in $\mathcal{G}$ under operations (a) and (b). Specifically, we show that one can insert one or four new squares corresponding to these operations in a rectangular gap of suitable size.

▶ **Lemma 2.** *For every $\alpha, \beta > 0$, there exists an axis-aligned rectangle that can be subdivided by two horizontal (resp., vertical) lines into three rectangles of aspect ratios α, 1, and β, respectively.*

Proof. Let R be a rectangle of aspect ratio $\alpha + \beta + 1$, with width x and height $(\alpha + \beta + 1)x$. Two horizontal lines at distance αx and βx from the top and bottom side of R, resp., subdivide R into rectangles of aspect ratios α, 1, and β, as required; see Fig. 2. ◀

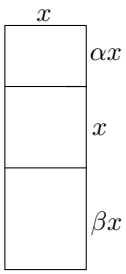


Figure 2 Constructing an outer rectangle given two inner rectangle aspect ratios.

To establish Theorem 1, we need a stronger version of Lemma 2 that allows the aspect ratios to vary within a small threshold.

▶ **Lemma 3.** *For every $\alpha, \beta, \varepsilon > 0$, there exists a $\delta > 0$ such that any rectangle of aspect ratio γ with $|\gamma - (\alpha + \beta + 1)| < \delta$ can be subdivided by two horizontal lines into rectangles of aspect ratios α', 1, and β' such that $|\alpha' - \alpha| < \varepsilon$ and $|\beta' - \beta| < \varepsilon$.*

Proof. Let $\delta = \min\{\alpha, \beta, 1, \varepsilon\}$. Let R be a rectangle of aspect ratio γ, where $|\gamma-(\alpha+\beta+1)| < \delta$, width x and height γx. Two horizontal lines at distance αx and $(1+\alpha)x$, resp., from the top side of R, subdivide R into rectangles of aspect ratios α, 1, and $\beta' = \gamma - \alpha - 1$. Note that $\beta' > 0$ and $|\beta' - \beta| = |\gamma - (\alpha + \beta + 1)| < \delta \leq \varepsilon$. ◀

▶ **Lemma 4.** *For every $\alpha_1, \ldots, \alpha_5 > 0$, there exists an axis-aligned rectangle R that can be subdivided into four squares and five rectangular gaps of aspect ratios $\alpha_1, \ldots, \alpha_5$ such that (refer to Figs. 1b and 5)*

- *the four squares are each in contact with a side of R, and their contact graph is a 4-cycle (but the contacts along the 4-cycle are not necessarily proper);*
- *the first four gaps are each incident to the upper-left, lower-left, lower-right, and upper-right corner of R, respectively, and the fifth gap lies in the interior of R.*

The proof of Lemma 4 requires some preparation, and is presented later in this section. The following lemma shows that all improper contacts can be replaced by proper contacts at the expense of allowing the five aspect ratios to vary within a given threshold. Using exact values of the aspect ratios, Lemma 4 can only guarantee single-point contacts. Figure 3 shows an example of aspect ratios which cannot be realized with proper contacts.

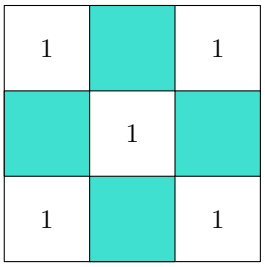


Figure 3 If all the gaps have aspect ratio 1, then scaling any of the squares to changing the point contacts into proper contacts would change the aspect ratios of the outer gaps.

▶ **Lemma 5.** *For every $\alpha_1, \ldots, \alpha_5 > 0$ and $\varepsilon > 0$, there exists a $\lambda > 0$ and a $\delta > 0$ such that every axis-aligned rectangle R of aspect ratio λ', $|\lambda - \lambda'| < \delta$, can be subdivided into four squares and five gaps of aspect ratios α_i', with $|\alpha_i' - \alpha_i| < \varepsilon$, for $i = 1, \ldots, 5$ such that*

- *the four squares are each in contact with a side of R, and their contact graph is a 4-cycle, and all contacts are proper;*
- *the first four gaps are each incident to the upper-left, lower-left, lower-right, and upper-right corner of R, respectively, and the fifth gap lies in the interior of R.*

Omitted proofs (including the proof of Lemma 5) are in the full version of the paper [10].

For convenience, we will rename $\alpha_1, \ldots, \alpha_5$ respectively based on the positions of the gaps to which they correspond as α_c (center), $\alpha_{t\ell}$ (top-left), α_{tr} (top-right), α_{br} (bottom-right), $\alpha_{b\ell}$ (bottom-left). Also, name the squares incident to the top, bottom, right, and left side of R as s_t, s_b, s_r, and s_ℓ, respectively.

We will prove Lemma 4 by starting with an initial configuration (Fig. 4), where the aspect ratio of the center gap is already α_c, and there are improper contacts between adjacent squares of the cycle. Then we incrementally modify the configuration, while the center gap

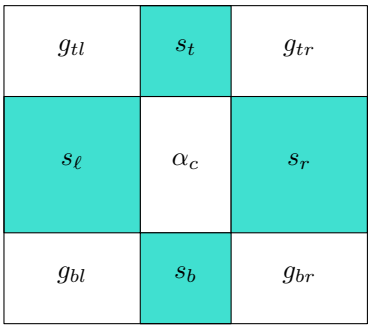
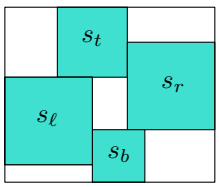
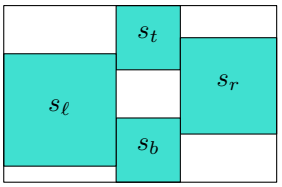


Figure 4 The initial configuration, with squares and gap aspect ratios labeled.

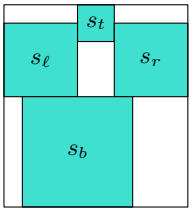
remains fixed, until all remaining gaps have the target aspect ratios $\alpha_{t\ell}$, α_{tr}, α_{br}, and $\alpha_{b\ell}$. We denote the current aspect ratios of these gaps by $g_{t\ell}$, g_{tr}, g_{br}, and $g_{b\ell}$ in the same fashion as $\alpha_{t\ell}$,...,$\alpha_{b\ell}$. In the full paper [10], we formally define four additional special configurations (shown in Fig. 5) that play a role in the incremental construction. Lemmas 6–10 below concern transformations of these configurations, and are used in the proof of Lemma 4.



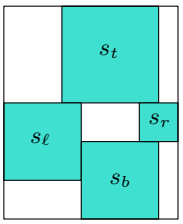
a) Clockwise Pinwheel



b) Vertical Stacked



c) Downward Arrow with directional square s_b



d) Clockwise Near-Pinwheel with reversed contact between s_t and s_r

Figure 5 Examples of four special configurations.

▶ **Lemma 6.** *Assume that the top-left corner of s_r is on the right side of s_t and the bottom-left corner of s_r is on the right side of s_b, and let $\alpha_{tr} > g_{tr}$ be given. There exists a $d > 0$ such that if we slide s_r upward by d and scale it up by a factor of d/g_{br} from its bottom-left corner, then no aspect ratio other than g_{tr} changes, and after the transformation we have $\alpha_{tr} = g_{tr}$, or $\alpha_{tr} > g_{tr}$ and s_r and s_b have a point contact. Similar statements hold after reflections and rotations of the configuration.*

Proof. Let the bottom-right gap have height h_1 and width w prior to the transformation. Assume that we slide s_r upward by some amount $d > 0$ and scale it up by a factor of d/g_{br} from its bottom-left corner After the transformation, it has height $h_1 + d$ and width $w + \frac{dw}{h_1}$.

As

$$\frac{h_1}{w} = \frac{h_1 + d}{w + \frac{dw}{h_1}},$$

the aspect ratio of the bottom-right gap has not changed. Let the height of top-right gap be h_2 prior to the transformation, and note that its width is also w. After the transformation, it has height $h - d$ and width $w + \frac{d}{g_{br}}$. Thus, its height monotonically decreases in d, and its width monotonically increases in d, so g_{tr} monotonically decreases in d. We can choose $d = \min(d_1, d_2)$, where $d_1 \geq 0$ is the value which would reduce the contact between s_r and s_b to a single point after the transformation, and $d_2 \geq 0$ is the value which would achieve $\alpha_{tr} = g_{tr}$. ◀

▶ **Lemma 7.** *A clockwise (resp., counterclockwise) pinwheel configuration can be transformed such that g_{br} or $g_{t\ell}$ (resp., g_{tr} or $g_{b\ell}$) increases to, or such that g_{tr} or $g_{b\ell}$ (resp., g_{br} or $g_{t\ell}$) decreases to any amount $\gamma > 0$, while all other aspect ratios remain the same.*

▶ **Lemma 8.** *A vertical (resp., horizontal) stacked configuration with a point contact between two of the squares can be transformed such that the aspect ratio of the outer gap between those squares increases (resp., decreases) to any amount $\gamma > 0$ while all other aspect ratios remain the same.*

▶ **Lemma 9.** *An upward or downward (resp., rightward or leftward) arrow configuration, with a point contact between the directional square and one of its neighbors, can be transformed such that the aspect ratio of the outer gap between those squares increases (resp., decreases) to any amount $\gamma > 0$ while all other aspect ratios remain the same.*

▶ **Lemma 10.** *A near-pinwheel configuration can be transformed such that the aspect ratio of the outer gap in the direction of the near-pinwheel (clockwise or counterclockwise) from the reversed contact increases to any amount $\gamma > 0$ if its left side is the side of a square, or decreases to any amount $\gamma > 0$ if its top side is the side of a square, while all other aspect ratios remain the same.*

Proof of Lemma 4. Let α_c, $\alpha_{t\ell}$, α_{tr}, α_{br}, and $\alpha_{b\ell}$ be given. Start with the initial configuration (cf. Fig. 4). If the target aspect ratios of all four outer gaps are α_c^{-1}, then R can be drawn now with aspect ratio α_c. Otherwise, one or more of the outer gaps must have their aspect ratios changed, either by increasing or decreasing them.

Rotate and reflect the initial configuration if necessary such that at least one gap needs to be made wider (i.e., $\alpha < g$), and the ratio g/α is maximal for the upper-right gap. In order to change g_{tr} to α_{tr}, we can scale up s_r from its lower-left corner until $g_{tr} = \alpha_{tr}$. This scaling will not affect $g_{t\ell}$ or $g_{b\ell}$, but it will decrease g_{br}. After the scaling, the bottom-right gap will either have the target aspect ratio already, need to be wider yet, or need to be narrower. From now on, we will not mention the case where a gap has reached its target aspect ratio already, because it just means that the next step can be skipped.

If the bottom-right gap needs to be wider yet, then by Lemma 6 we can scale up s_r and translate it downward until $g_{br} = \alpha_{br}$ without changing g_{tr}. As g/α is assumed to be maximal for the upper-right gap, if this transformation results in a point contact between s_r and s_t, it also achieves $g_{br} = \alpha_{br}$ (because otherwise, $g_{br} > g_{tr} = \alpha_{tr}$).

If the bottom-right gap needs to be narrower, then we can scale up s_b from its top-left corner until $g_{br} = \alpha_{br}$. This will increase $g_{b\ell}$.

Now, we can assume that $g_{tr} = \alpha_{tr}$ and $g_{br} = \alpha_{br}$. We distinguish between four cases:

1. s_b has not been scaled, and either $\alpha_{b\ell} \leq \alpha_c^{-1}$ or $\alpha_{t\ell} \leq \alpha_c^{-1}$.
2. s_b has been scaled up from its top-left corner, $\alpha_{b\ell} \leq \alpha_{t\ell}$, and $\alpha_{b\ell} \leq \alpha_c^{-1}$.
3. s_b has been scaled up from its top-left corner, $\alpha_{t\ell} \leq \alpha_{b\ell}$, and $\alpha_{t\ell} \leq \alpha_c^{-1}$.
4. $\alpha_{t\ell} > \alpha_c^{-1}$ and $\alpha_{b\ell} > \alpha_c^{-1}$.

Case 1: s_b has not been scaled, and either $\alpha_{b\ell} \leq \alpha_c^{-1}$ or $\alpha_{t\ell} \leq \alpha_c^{-1}$. Reflect the configuration such that $\alpha_{b\ell} \leq \alpha_{t\ell}$. Scale up s_ℓ from its top-right corner until $g_{b\ell} = \alpha_{b\ell}$ (making the top-left gap wider). Then, if $g_{t\ell}$ needs to decrease further, by Lemma 6 we can scale up and translate s_ℓ until $g_{t\ell} = \alpha_{t\ell}$ to achieve all target aspect ratios (once again, this transformation guarantees $g_{t\ell} = \alpha_{t\ell}$ even if it results in a point contact, because we assume $\alpha_{b\ell} \leq \alpha_{t\ell}$). Otherwise, the top-left gap needs to be narrower. Since the configuration is a horizontal stacked configuration, and by Lemma 8 we can apply a series of transformations to achieve all target aspect ratios.

Case 2: s_b has been scaled up from its top-left corner, $\alpha_{b\ell} \leq \alpha_{t\ell}$, and $\alpha_{b\ell} \leq \alpha_c^{-1}$. Scale up s_ℓ from its top-right corner until $g_{b\ell} = \alpha_{b\ell}$. This transformation decreases $g_{t\ell}$. Then, if $g_{t\ell}$ needs to decrease further, by Lemma 6 we can scale up and translate s_ℓ until $g_{t\ell} = \alpha_{t\ell}$ to achieve all target aspect ratios (once again guaranteed because $\alpha_{b\ell} \leq \alpha_{t\ell}$). Otherwise the top-left gap needs to be narrower. Since the squares are arranged in a pinwheel configuration, Lemma 7 completes the proof.

Case 3: s_b has been scaled up from its top-left corner, $\alpha_{t\ell} \leq \alpha_{b\ell}$, and $\alpha_{t\ell} \leq \alpha_c^{-1}$. Scale up s_ℓ from its bottom-right corner until $g_{t\ell} = \alpha_{t\ell}$. This transformation decreases $g_{b\ell}$. Then, if $g_{b\ell}$ needs to decrease further, by Lemma 6 we can scale up s_ℓ and translate it downward, maintaining all other aspect ratios, until $g_{b\ell} = \alpha_{b\ell}$ or s_ℓ and s_t have a point contact. If s_ℓ and s_t have a point contact, then the squares are arranged in a pinwheel configuration, and Lemma 7 completes the proof. Otherwise, $g_{b\ell}$ needs to increase. Since the squares form a downward arrow configuration in this case, with a point contact between s_b and s_ℓ, Lemma 9 completes the proof.

Case 4: $\alpha_{t\ell} > \alpha_c^{-1}$ and $\alpha_{b\ell} > \alpha_c^{-1}$. We distinguish between two subcases.

Case 4.1: If the top-right corner of s_b lies on the bottom side of s_r, then by Lemma 6, we can translate s_b to the left while scaling it up until $g_{b\ell} = \alpha_{b\ell}$ or s_b and s_r have a point-contact, while maintaining all other aspect ratios. If $g_{b\ell} = \alpha_{b\ell}$, then the configuration is a near-pinwheel and Lemma 10 completes the proof. Otherwise, if s_b and s_r have a point-contact, then the conditions of Case 4.2 below are satisfied and we proceed as follows.

Case 4.2: If the top-right corner of s_b lies on the left side of s_r, then scale up s_t from its bottom-right corner until $g_{t\ell} = \alpha_{t\ell}$ and scale up s_b from its top-right corner until $g_{b\ell} = \alpha_{b\ell}$. Now, g_{tr} and g_{br} (which were previously at their target values) both need to decrease. Reflect the configuration if necessary so that the width of the bottom-right gap needs to be increased by a larger amount than the top-right gap. Scale up s_r from its lower-left corner until $g_{tr} = \alpha_{tr}$. Then, because the width of the lower-right gap needed to be increased by a larger amount of the two, it still needs to be wider. The configuration is a rightward arrow, so by Lemma 9, we can decrease g_{br} arbitrarily while maintaining the other aspect ratios. ◀

It is easy to extend Lemma 4 to Lemma 5 by changing any improper contacts among adjacent squares in the 4-cycle into proper contacts. Theorem 1 then follows by induction. See the full paper [10] for the proof of Lemma 5 and Theorem 1.

References

1 Melanie Badent, Carla Binucci, Emilio Di Giacomo, Walter Didimo, Stefan Felsner, Francesco Giordano, Jan Kratochvíl, Pietro Palladino, Maurizio Patrignani, and Francesco

Trotta. Homothetic triangle contact representations of planar graphs. In *Proc. 19th Canadian Conference on Computational Geometry (CCCG)*, pages 233–236, Ottawa, ON, Canada, 2007. Carleton University. URL: http://cccg.ca/proceedings/2007/09b4.pdf.

2 Giordano Da Lozzo, William E. Devanny, David Eppstein, and Timothy Johnson. Square-contact representations of partial 2-trees and triconnected simply-nested graphs. In *Proc. 28th Symposium on Algorithms and Computation (ISAAC)*, volume 92 of *LIPIcs*, pages 24:1–24:14. Schloss Dagstuhl, 2017. doi:10.4230/LIPIcs.ISAAC.2017.24.

3 Hubert de Fraysseix, Patrice Ossona de Mendez, and Pierre Rosenstiehl. On triangle contact graphs. *Comb. Probab. Comput.*, 3:233–246, 1994. doi:10.1017/S0963548300001139.

4 David Eppstein. Square contact graphs, 2017. URL: https://11011110.github.io/blog/2017/10/03/square-contact-graphs.html.

5 Stefan Felsner and Mathew C. Francis. Contact representations of planar graphs with cubes. In *Proc. 27th Symposium on Computational Geometry (SoCG)*, pages 315–320. ACM Press, 2011. doi:10.1145/1998196.1998250.

6 Daniel Gonçalves, Benjamin Lévêque, and Alexandre Pinlou. Triangle contact representations and duality. *Discret. Comput. Geom.*, 48(1):239–254, 2012. doi:10.1007/s00454-012-9400-1.

7 Jonathan Klawitter, Martin Nöllenburg, and Torsten Ueckerdt. Combinatorial properties of triangle-free rectangle arrangements and the squarability problem. In *Proc. 23rd Symposium on Graph Drawing and Network Visualization (GD)*, volume 9411 of *LNCS*, pages 231–244. Springer, 2015. doi:10.1007/978-3-319-27261-0_20.

8 Paul Koebe. Kontaktprobleme der Konformen Abbildung. *Ber. Sächs. Akad. Wiss. Leipzig, Math.-Phys. Kl.*, 88:141–164, 1936.

9 László Lovász. *Graphs and Geometry*. 2019. URL: http://web.cs.elte.hu/~lovasz/bookxx/geomgraphbook/geombook2019.01.11.pdf.

10 Andrew Nathenson. Axis-aligned square contact representations. *Preprint*, 2021. arXiv:2103.08719.

11 Oded Schramm. Existence and uniqueness of packings with specified combinatorics. *Israel Journal of Mathematics*, 73:321–341, 1991. doi:10.1007/BF02773845.

The Voronoi Diagram of Rotating Rays with applications to Floodlight Illumination

Carlos Alegría[1], Ioannis Mantas[2], Evanthia Papadopoulou[2], Marko Savić[3], Hendrik Schrezenmaier[4], Carlos Seara[5], and Martin Suderland[2]

1 **Dipartimento di Ingegneria, Università Roma Tre, Rome, Italy**
carlos.alegria@uniroma3.it
2 **Faculty of Informatics, Università della Svizzera italiana, Lugano, Switzerland**
{ioannis.mantas, evanthia.papadopoulou, martin.suderland}@usi.ch
3 **Department of Mathematics and Informatics, Faculty of Sciences, University of Novi Sad, Novi Sad, Serbia**
marko.savic@dmi.uns.ac.rs
4 **Institut für Mathematik, Technische Universität Berlin, Berlin, Germany**
schrezen@math.tu-berlin.de
5 **Departament de Matemàtiques, Universitat Politècnica de Catalunya, Barcelona, Spain.**
carlos.seara@upc.edu

Abstract

We introduce the *Rotating Rays Voronoi diagram*, a Voronoi structure where the input sites are rays and the distance function is the counterclockwise angular distance. This novel diagram can be used to solve illumination or coverage problems where a domain has to be covered by floodlights/wedges of uniform angle. We present structural properties, combinatorial complexity bounds, and algorithms to construct the diagram. Moreover, we show how we can use this Voronoi diagram to compute the *Brocard angle* of a convex polygon in optimal linear time.

1 Introduction

In this work, we study a Voronoi diagram where the input is a set of n rays in the plane, and the distance from a point x to a ray r is the angular distance, i.e., the minimum angle α such that, after counterclockwise rotating r around its apex by α, r *illuminates* x; see Fig. 1.

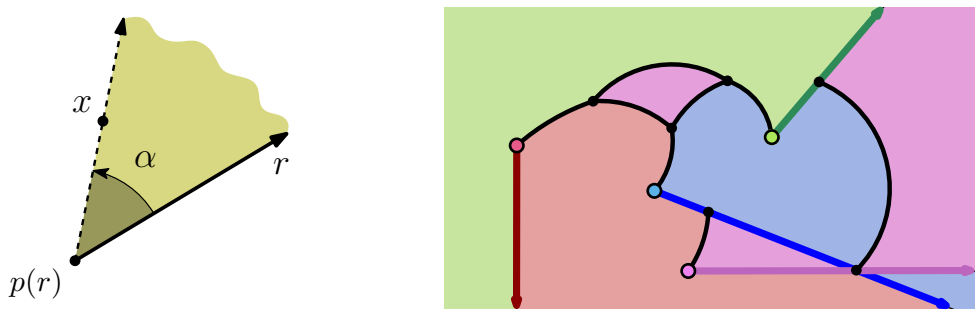
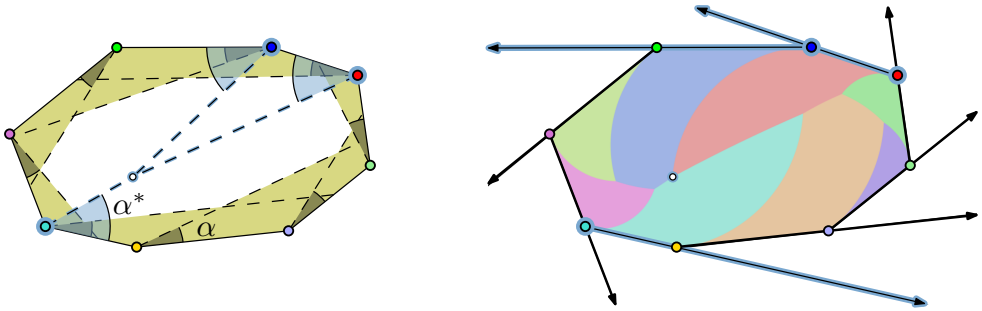


Figure 1 (a) The angular distance α from a point x to a ray r. (b) The Voronoi diagram of four rays in the plane. All points in a region are first illuminated by the ray with the respective color.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.



(a) (b)

Figure 2 (a) Illumination with edge-aligned α-floodlights. (b) The Rotating Rays Voronoi diagram. Highlighted are the point and the three rays that realize the Brocard angle α^*.

Motivation. The distance measure is motivated by the following illumination problem: An *α-floodlight* is a light source that illuminates a cone with aperture α from its apex. Given a simple polygon P, an α-floodlight is placed on each vertex $v \in P$ facing the interior of P in such a way, that one of its rays contains the successor of v in the counterclockwise order of the vertices of P; see Fig. 2a. The *Brocard Illumination problem* [1] asks for the *Brocard angle*, the smallest value of α for which a set of α-floodlights covers the interior of P.

Constructing the Voronoi diagram of rays inside a convex polygon reveals the Brocard angle, as the latter is realized at a vertex of the diagram with maximum distance; see Fig. 2b. Interestingly, given a set of rays, similar illumination problems can be defined in different domains. The domain may be the entire plane or even a curve and, analogously, constructing the Voronoi diagram in that domain yields the Brocard angle. Hence, there is an interest in studying such diagrams, and designing construction algorithms for different domains.

Our contribution. We introduce the *Rotating Rays Voronoi Diagram* and prove a series of results, paving the way for future work on similar problems. We consider the diagram in the plane and identify structural properties which we complement with combinatorial complexity results, a worst case $\Omega(n^2)$ lower bound and an $O(n^{2+\epsilon})$ upper bound, together with an $O(n^{2+\epsilon})$-time construction algorithm. Finally, motivated by applications to illumination problems, we restrict our domain to a convex polygonal region bounded by the input set of sites, and construct the Voronoi diagram in such a domain in optimal $\Theta(n)$ time.

Related work. In the Brocard illumination problem, a polygon is called a *Brocard polygon* [2] if all the α-floodlights simultaneously illuminate a point inside the polygon when α is equal to the Brocard angle. The characterization of Brocard polygons has a long history, yet, only harmonic polygons (which include triangles and regular polygons) are known to be Brocard [5]. Deciding whether a polygon is Brocard can be done in $O(n)$ time and then the Brocard angle can be computed in $O(1)$ time. Computing the Brocard angle of simple polygons was first studied by Alegría et al. [1]. The authors solved the problem in $O(n^3 \log^2 n)$ time, and complemented this result with an $O(n \log n)$-time algorithm for convex polygons.

Since their introduction [4], floodlight illumination problems have been widely studied, see e.g. [15, 20]. The case when the floodlights are of uniform angle is of particular interest, and has been explored by several authors, see e.g. [6, 9, 10, 16, 19]. Rotating α-floodlights are also used to model devices with limited sensing range, like surveillance cameras or directional antennae [7, 13, 14]. The Rotating Rays Voronoi diagram seems to be novel with respect to both the input sites and the distance function. A related diagram was defined [8] to study dominance regions of players in the analysis of football (soccer) matches [18].

2 Preliminaries

In this section we formally define the Rotating Rays Voronoi diagram. Let $\mathcal{S}$ be a set of n rays in the plane. Given a ray r, we denote its apex by $p(r)$ and its supporting line by $l(r)$.

▶ **Definition 2.1.** Given a ray r and a point $x \in \mathbb{R}^2$, the *angular distance* from x to r, denoted $d_\angle(x, r)$, is the minimum counterclockwise angle α from r to a ray with apex on $p(r)$ passing through x; see Fig. 1a.

▶ **Definition 2.2.** Given two rays r and s, the *dominance region* $dr(r, s)$ is the set of points with smaller angular distance to r than to s, i.e., $dr(r, s) = \{x \in \mathbb{R}^2 \mid d_\angle(x, r) < d_\angle(x, s)\}$. The *angular bisector* of r and s, denoted $b_\angle(r, s)$, is the curve delimiting $dr(r, s)$ and $dr(s, r)$.

The different types of bisectors are illustrated in Fig. 3. Given two rays r and s, let $I = l(r) \cap l(s)$. The bisector $b_\angle(r, s)$ is the union of r, s, and a circular arc a that connects $p(r)$ to $p(s)$. The arc a belongs to the *bisecting circle* $C_b(r, s)$, which we define as follows:

- If I, $p(r)$, and $p(s)$ are pairwise different, then $C_b(r, s)$ is the circle through I, $p(r)$, and $p(s)$. The arc a contains I if, and only if, I lies either on none or on both of r and s.
- If $I = p(r)$ and $I \neq p(s)$, then $C_b(r, s)$ is the circle tangent to $l(r)$ passing through $p(r)$ and $p(s)$. Both a and r lie on the same side of $l(s)$ if, and only if, $p(r)$ lies on s. We analogously define $C_b(r, s)$ if $I = p(s)$ and $I \neq p(r)$.
- If $p(r) = p(s)$, then both $C_b(r, s)$ and a degenerate to a single point.
- If $l(r)$ and $l(s)$ are parallel, then $C_b(r, s)$ degenerates to the line through $p(r)$ and $p(s)$, and a degenerate either to a line segment or to two halflines.

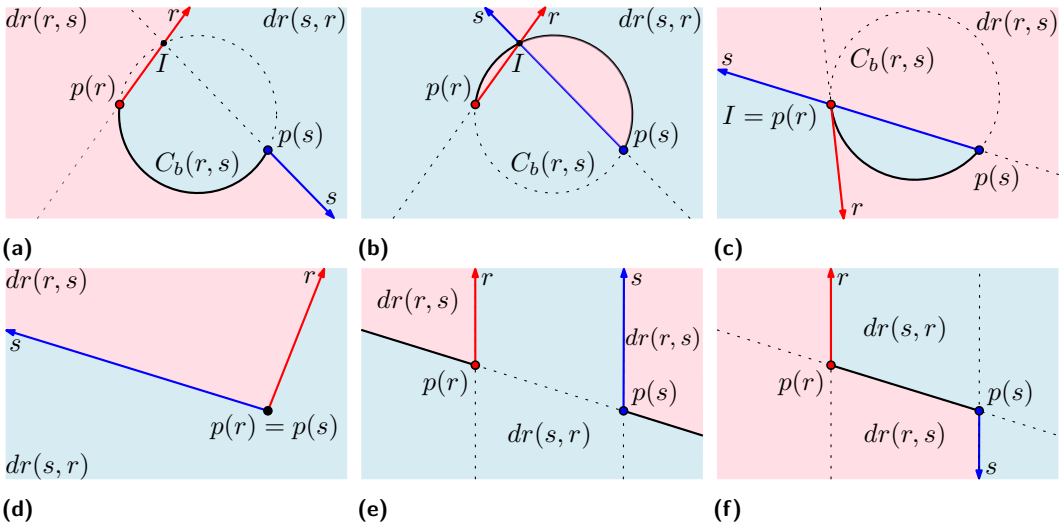


Figure 3 The bisector of two rays r and s which are: (a) non-intersecting, (b) intersecting, (c) with $p(r)$ lying on s, (d) sharing their apex, (e) parallel, and (f) anti-parallel.

▶ **Definition 2.3.** The *Rotating Rays Voronoi diagram* of a set $\mathcal{S}$ of rays is the subdivision of $\mathbb{R}^2$ into *nearest Voronoi regions* defined as follows:

$$vreg(r) := \{\, x \in \mathbb{R}^2 \mid \forall s \in \mathcal{S} \setminus \{r\} :\ d_\angle(x, r) < d_\angle(x, s) \,\}.$$

The graph structure of the diagram of $\mathcal{S}$ is denoted by $\mathrm{RVD}(\mathcal{S}) := \left(\mathbb{R}^2 \setminus \bigcup_{r \in \mathcal{S}} vreg(r)\right) \cup \mathcal{S}$.

An example diagram is shown in Fig. 1b. A region $vreg(r)$ can be equivalently defined as the intersection of all the dominance regions of r, i.e., $vreg(r) = \bigcap_{s \in \mathcal{S} \setminus \{r\}} dr(r, s)$. Note that a region can have more than one connected component, which we call a *face* of the region.

3 RVD: properties, complexity, and an algorithm

In this section we study the diagram in the plane. Assuming that no two rays of $\mathcal{S}$ are parallel to each other, the following two structural properties hold.

▶ **Lemma 3.1.** *RVD($\mathcal{S}$) has exactly n unbounded faces, one for each ray.*

Proof sketch. Let Γ be a circle containing all the bisecting circles of $\mathcal{S}$. The intersection of Γ and the unbounded part of each dominance region of $r \in \mathcal{S}$ is a circular arc with endpoint $r \cap \Gamma$. The intersection of all such arcs is connected. Hence $vreg(r) \cap \Gamma$ is connected. ◀

▶ **Lemma 3.2.** *RVD($\mathcal{S}$) is connected.*

Proof sketch. If RVD($\mathcal{S}$) is not connected, then there is a region $vreg(r)$ that either has an unbounded face with two occurrences at infinity or it encloses a component of RVD($\mathcal{S}$), creating an *"island"*. The first case is excluded by Lemma 3. The second case implies that such an island also exists in some dominance region of the site r, which leads to a contradiction. ◀

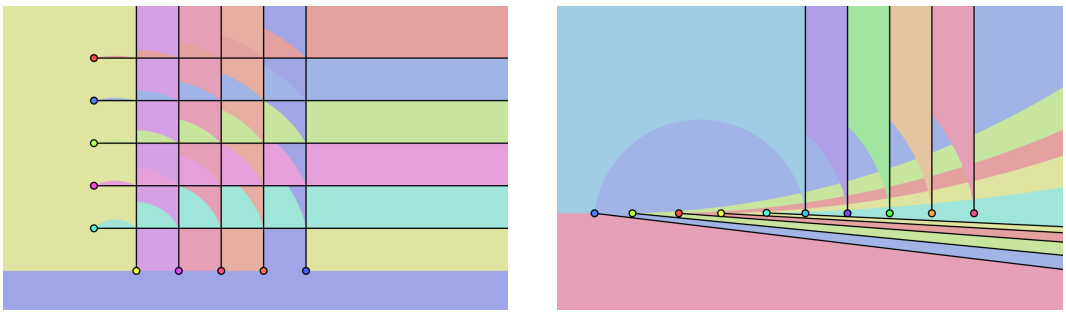
We now show two lower bound constructions for the worst case complexity of the RVD.

▶ **Theorem 3.3.** *RVD($\mathcal{S}$) has $\Omega(n^2)$ combinatorial complexity in the worst case. This holds even if the rays are pairwise non-intersecting.*

Proof sketch. The bound is achieved by creating a grid structure in which the rays have $\Theta(n^2)$ intersections, each inducing a vertex in RVD($\mathcal{S}$); see Figure 4a. A $\Theta(n^2)$ construction can also be achieved by a set of rays that are pairwise disjoint. An example is shown in Figure 4b, where the Voronoi regions of the $n/2-1$ leftmost rays have $n/2+1$ faces each. ◀

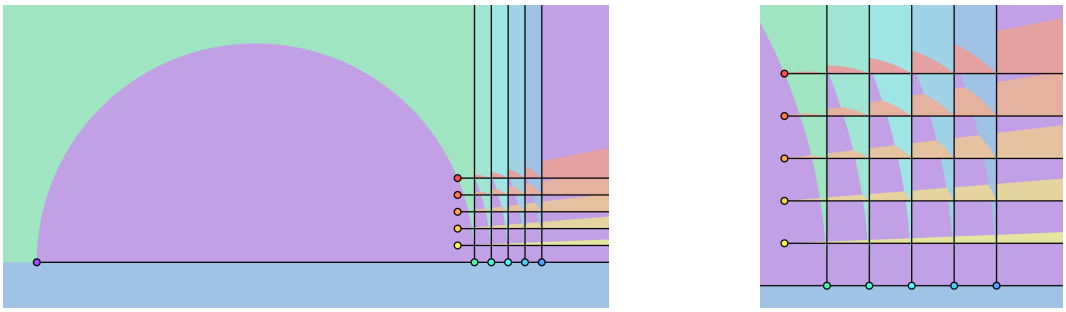
▶ **Lemma 3.4.** *A Voronoi region of RVD($\mathcal{S}$) has $\Theta(n^2)$ complexity in the worst case.*

Proof sketch. Any vertex incident to a region $vreg(r)$, is defined by r and a pair of rays. Further, the diagram of three rays has $O(1)$ complexity, so the $O(n^2)$ bound follows. For the lower bound, we create a grid structure as in Theorem 3.3. We then add a ray r so that $vreg(r)$ has a face inside each cell of the grid, thus $vreg(r)$ has $\Theta(n^2)$ complexity; see Fig. 5. ◀



(a) **(b)**

Figure 4 RVD($\mathcal{S}$) with $\Theta(n^2)$ complexity, where the rays in $\mathcal{S}$ are (a) intersecting and (b) pairwise non-intersecting.



(a) **(b)**

Figure 5 RVD($\mathcal{S}$) with a region having $\Theta(n^2)$ complexity (a) zoomed out and (b) zoomed in.

Finally, we apply the general upper bound by Sharir [17] to yield a near quadratic upper bound on the complexity of the RVD. This is accompanied by a construction algorithm.

▶ **Theorem 3.5.** *For any $\epsilon > 0$, RVD($\mathcal{S}$) has $O(n^{2+\epsilon})$ combinatorial complexity. Further, RVD($\mathcal{S}$) can be constructed in $O(n^{2+\epsilon})$ time.*

Proof sketch. Each site induces a distance function which maps each point in the plane to its distance to that site. The RVD can be seen as a minimization diagram of these distance functions. With a monotone increasing transformation we map these distance functions to algebraic functions. Applying the transformations keeps the lower envelope of the distance functions invariant. The lower envelope of these n algebraic functions has $O(n^{2+\epsilon})$ combinatorial complexity and it can be constructed in $O(n^{2+\epsilon})$ time [17]. ◀

We can extend the Brocard Illumination problem to the plane as follows. We place an α-floodlight f_r on every $r \in \mathcal{S}$ such that f_r is equal to r when $\alpha = 0$. We want the minimum angle α^* for which the set $\{f_r \mid r \in \mathcal{S}\}$ of α^*-floodlights illuminates the plane. The angle α^* is realized at a point of maximum distance, which is either a vertex of RVD($\mathcal{S}$) or a point at infinity on a ray of $\mathcal{S}$. Hence, we can find α^* by constructing RVD($\mathcal{S}$) and then traversing the diagram in linear time in its size. Note that α^* takes values in the interval $(2\pi/n, 2\pi)$.

4 RVD of a convex polygon: Brocard illumination

We now turn our attention to the Brocard illumination problem. Given a convex polygon P with n vertices, we describe an algorithm to compute the Brocard angle α^* of P by means of the RVD. Let $\mathcal{S}_P$ be the set of n rays such that each ray has a vertex $v \in P$ as apex, and passes through the successor of v in the counterclockwise order of the vertices of P. Let PRVD($\mathcal{S}_P$) be the part of RVD($\mathcal{S}_P$) restricted to the interior of P. Note that PRVD($\mathcal{S}_P$) has $\Theta(n)$ complexity, as opposed to RVD($\mathcal{S}_P$) which can have $\Theta(n^2)$. We show the following.

▶ **Theorem 4.1.** *Given a convex polygon P, we can construct PRVD($\mathcal{S}_P$) in $\Theta(n)$ time. The Brocard angle of P can also be found in the same time.*

Algorithm outline. We first partition $\mathcal{S}_P$ into four sets $\mathcal{S}_N, \mathcal{S}_W, \mathcal{S}_S$ and $\mathcal{S}_E$ of consecutive rays, such that any two rays in a subset have an angular difference at most $\pi/2$, see Fig. 6a. For each set $\mathcal{S}_d$, $d \in \{\text{N,W,S,E}\}$, we obtain a set $\mathcal{S}_d^r$ in which every ray of $\mathcal{S}_d$ is rotated by an angle of $-\pi/2$. Then, we construct each diagram RVD($\mathcal{S}_d^r$) independently, see Fig. 6c. Finally, we merge the four diagrams in two steps to obtain PRVD($\mathcal{S}_P$), see Figs. 7 and 8.

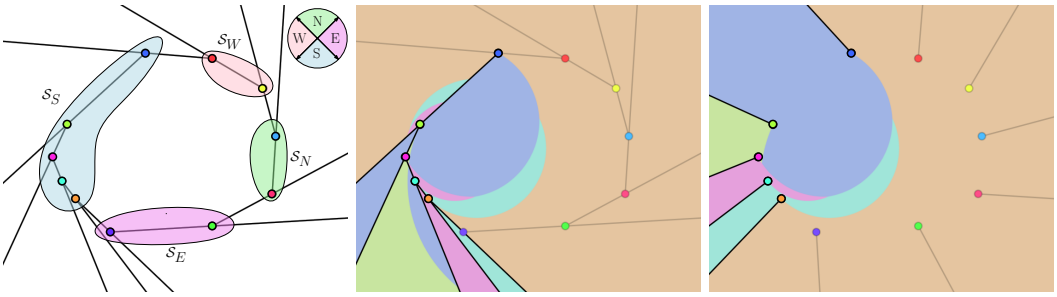


Figure 6 (a) Partition of $\mathcal{S}_P$ into sets $\mathcal{S}_N, \mathcal{S}_W, \mathcal{S}_S$ and $\mathcal{S}_E$. (b) RVD($\mathcal{S}_S$). (c) RVD($\mathcal{S}_S^r$)

Constructing the diagrams. To construct the diagram of each subset $\mathcal{S}_d^r$ in optimal $\Theta(|\mathcal{S}_d^r|)$ time, we make use of the *abstract Voronoi diagrams* framework [11, 12]. To fall under this framework, the underlying system of bisectors must satisfy the following three *axioms*:

A1 The bisector $b_\angle(r, s)$, $\forall r, s \in \mathcal{S}_d^r$, is an unbounded Jordan curve.
A2 The region $vreg(r)$ in RVD($\mathcal{S}'$), $\forall \mathcal{S}' \subseteq \mathcal{S}_d^r$ and $\forall r \in \mathcal{S}'$, is connected.
A3 The closure of the union of all regions in RVD($\mathcal{S}'$), $\forall \mathcal{S}' \subseteq \mathcal{S}_d^r$, covers $\mathbb{R}^2$.

▶ **Lemma 4.2.** *The system of bisectors of $\mathcal{S}_d^r$ satisfies the axioms A1-A3.*

Proof sketch. A1 is satisfied since the rays are disjoint. A2 holds due to the rotation of $-\pi/2$: after the rotation, no ray intersects twice any bisecting circle, thus no bounded faces appear, see e.g., Figs. 6b and 6c. A3 is satisfied as each ray induces a distance function. ◀

The intuition for the rotation is twofold: On PRVD($\mathcal{S}_P$) only circular parts of bisectors appear and under $-\pi/2$ rotation the circular parts of the bisectors remain the same. Note that there does not exist a rotation angle for which the complete set $\mathcal{S}_P$ satisfies axiom A2.

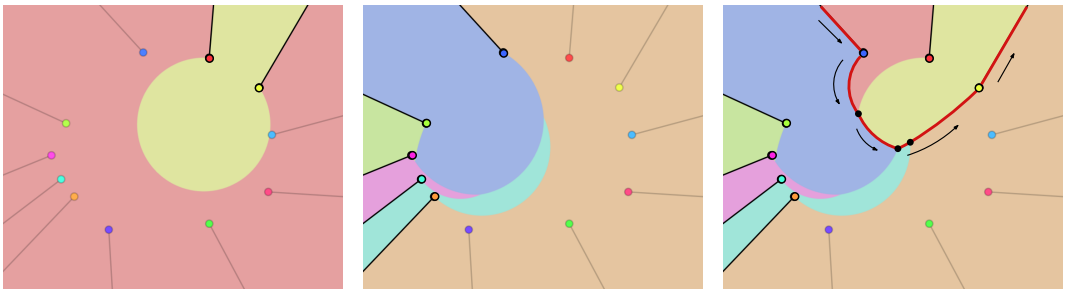
To construct RVD($\mathcal{S}_d^r$) we use the $\Theta(n)$-time algorithm of [3]. Apart from satisfying axioms A1-A3, [3] requires that the order of the regions of RVD($\mathcal{S}'$) along a simple curve is known, for any $\mathcal{S}' \subseteq \mathcal{S}_d^r$. Since, the rays are non-intersecting, this order coincides with the order of the rays of $\mathcal{S}'$ along the boundary of the polygon. We obtain the following lemma.

▶ **Lemma 4.3.** *RVD($\mathcal{S}_d^r$) is a tree of $\Theta(|\mathcal{S}_d^r|)$ complexity. Further, RVD($\mathcal{S}_d^r$) can be constructed in $\Theta(|\mathcal{S}_d^r|)$ time.*

Merging the diagrams. We merge all four diagrams, in two steps, to obtain PRVD($\mathcal{S}_P$). In a first step we merge RVD($\mathcal{S}_W^r$) with RVD($\mathcal{S}_S^r$) to obtain RVD($\mathcal{S}_W^r \cup \mathcal{S}_S^r$), where the *merge curve* consists of the first ray in each of $\mathcal{S}_W^r$ and $\mathcal{S}_S^r$ and the set of circular edges, which are equidistant to both sets; see Fig. 7. We respectively obtain RVD($\mathcal{S}_E^r \cup \mathcal{S}_N^r$). Then, in a second step we merge the diagrams RVD($\mathcal{S}_W^r \cup \mathcal{S}_S^r$) and RVD($\mathcal{S}_E^r \cup \mathcal{S}_N^r$), restricted to the interior of P, to obtain PRVD($\mathcal{S}_P$); see Fig. 8. Using standard tracing techniques for Voronoi diagrams and the properties of the angular bisectors, we can prove the following result.

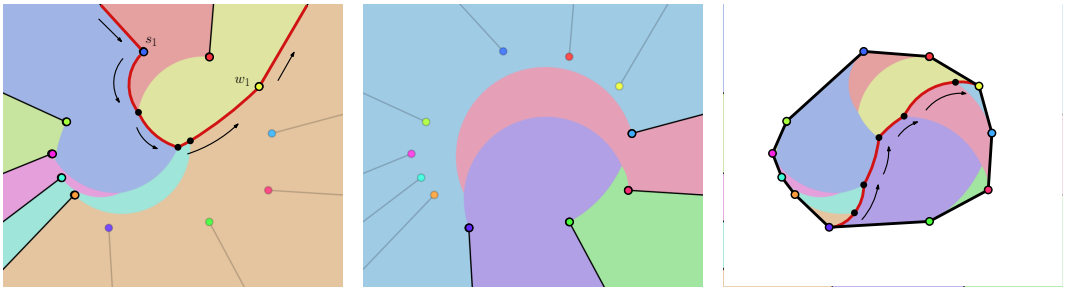
▶ **Lemma 4.4.** *Given RVD($\mathcal{S}_d^r$) $\forall d \in$ {N,W,S,E}, PRVD($\mathcal{S}_P$) can be constructed in $\Theta(n)$ time.*

Concluding this section, the Brocard angle of P, denoted by α^*, is realized at a point of maximum distance in the interior of P which lies on a vertex of PRVD($\mathcal{S}_P$). Hence, we can solve the Brocard Illumination problem in $\Theta(n)$ time, by constructing PRVD($\mathcal{S}_P$) and then traversing it, to obtain α^*. Note that α^* takes values in the interval $(0, \pi/2 - \pi/n]$.



(a) (b) (c)

Figure 7 Merging diagrams (a) RVD($\mathcal{S}_W^r$) and (b) RVD($\mathcal{S}_S^r$) into (c) RVD($\mathcal{S}_W^r \cup \mathcal{S}_S^r$).



(a) (b) (c)

Figure 8 Merging diagrams (a) RVD($\mathcal{S}_W^r \cup \mathcal{S}_S^r$) and (b) RVD($\mathcal{S}_E^r \cup \mathcal{S}_N^r$) into (c) PRVD($\mathcal{S}_P$).

5 Concluding remarks

By means of the Rotating Rays Voronoi diagram, we showed how to find in optimal time the Brocard angle of a convex polygon, settling an interesting geometric problem. Our method is more general: given any domain D and a set of rays $\mathcal{S}$, we can find the minimum angle needed to illuminate D with floodlights aligned at $\mathcal{S}$, by constructing RVD($\mathcal{S}$) inside D.

There are many questions to investigate. What is the worst case complexity of the diagram in the plane? Is it $\Theta(n^2)$? Can the diagram in the plane be constructed in time $o(n^{2+\epsilon})$? How does our approach to compute the Brocard angle extend to other classes of polygons?

Acknowledgments. Initial discussions took place at the Intensive Research Program in Discrete, Combinatorial and Computational Geometry in Barcelona, Spain, in 2018. We are grateful to the Centre de Recerca Matemàtica, Universitat Autònoma de Barcelona, for hosting the event and to the organizers for providing the platform to meet and collaborate.

C.A. was supported by MIUR Proj. "AHeAD" n° 20174LF3T8. Early work of I.M. and E.P. was supported by the DACH project Voronoi++, SNF-200021E_154387. M.S. is partly supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia (Grant No. 451-03-68/2020-14/200125), and the Provincial Secretariat for Higher Education and Scientific Research, Province of Vojvodina. Early work of H.S. was partially supported by the German Research Foundation, DFG grant FE-340/11-1. C.S. was supported by project PID2019-104129GB-I00/AEI/10.13039/501100011033. This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 734922.

References

1 Carlos Alegría-Galicia, David Orden, Carlos Seara, and Jorge Urrutia. Illuminating polygons by edge-aligned floodlights of uniform angle (Brocard illumination). In *Proc. European Workshop on Computational Geometry*, pages 281–284, 2017.

2 Arthur Bernhart. Polygons of pursuit. *Scripta Mathematica*, 24(1):23–50, 1959.

3 Cecilia Bohler, Rolf Klein, Andrzej Lingas, and Chih-Hung Liu. Forest-like abstract Voronoi diagrams in linear time. *Computational Geometry*, 68:134–145, 2018.

4 Prosenjit Bose, Leonidas J. Guibas, Anna Lubiw, Mark Overmars, Diane Souvaine, and Jorge Urrutia. The floodlight problem. *International Journal of Computational Geometry & Applications*, 7(1-2):153–163, 1997.

5 John Casey. *A sequel to the first six books of the elements of Euclid.* Dublin University Press., 1888.

6 Felipe Contreras, Jurek Czyzowicz, Nicolas Fraiji, and Jorge Urrutia. Illuminating triangles and quadrilaterals with vertex floodlights. In *Proc. Canadian Conference on Computational Geometry*, 1998.

7 Jurek Czyzowicz, Stefan Dobrev, Benson Joeris, Evangelos Kranakis, Danny Krizanc, Jan Maňuch, Oscar Morales-Ponce, Jaroslav Opatrny, Ladislav Stacho, and Jorge Urrutia. Monitoring the plane with rotating radars. *Graphs and Combinatorics*, 31(2):393–405, 2015.

8 Mark de Berg, Joachim Gudmundsson, Herman Haverkort, and Michael Horton. Voronoi diagrams with rotational distance costs. In *Computational Geometry Week: Young Researchers Forum*, 2017.

9 Vladimir Estivill-Castro, Joseph O'Rourke, Jorge Urrutia, and Dianna Xu. Illumination of polygons with vertex lights. *Information Processing Letters*, 56(1):9–13, 1995.

10 Dan Ismailescu. Illuminating a convex polygon with vertex lights. *Periodica Mathematica Hungarica*, 57(2):177–184, 2008.

11 Rolf Klein. *Concrete and Abstract Voronoi diagrams*, volume 400. Springer Science & Business Media, 1989.

12 Rolf Klein, Elmar Langetepe, and Zahra Nilforoushan. Abstract Voronoi diagrams revisited. *Computational Geometry: Theory and Applications*, 42(9):885–902, 2009.

13 Evangelos Kranakis, Danny Krizanc, and Oscar Morales. Maintaining connectivity in sensor networks using directional antennae. In *Theoretical Aspects of Distributed Computing in Sensor Networks*, pages 59–84. Springer, 2011.

14 Azin Neishaboori, Ahmed Saeed, Khaled A. Harras, and Amr Mohamed. On target coverage in mobile visual sensor networks. In *Proc. ACM International Symposium on Mobility Management and Wireless Access*, pages 39–46, 2014.

15 Joseph O'Rourke. Visibility. In *Handbook of discrete and computational geometry*, pages 875–896. CRC Press, 2017.

16 Joseph O'Rourke, Thomas Shermer, and Ileana Streinu. Illuminating convex polygons with vertex floodlights. In *Proc. Canadian Conference on Computational Geometry*, pages 151–156, 1995.

17 Micha Sharir. Almost tight upper bounds for lower envelopes in higher dimensions. *Discrete & Computational Geometry*, 12(3):327–345, 1994.

18 Tsuyoshi Taki, Jun-ichi Hasegawa, and Teruo Fukumura. Development of motion analysis system for quantitative evaluation of teamwork in soccer games. In *Proc. IEEE International Conference on Image Processing*, volume 3, pages 815–818. IEEE, 1996.

19 Csaba D. Tóth. Art galleries with guards of uniform range of vision. *Computational Geometry*, 21(3):185–192, 2002.

20 Jorge Urrutia. Art gallery and illumination problems. In *Handbook of Computational Geometry*, pages 973–1027. Elsevier, 2000.

The Fréchet Distance for Plane Graphs*

Pan Fang[1] and Carola Wenk†[1]

1 **Department of Computer Science, Tulane University, Louisiana, USA**
pfang@tulane.edu
2 **Department of Computer Science, Tulane University, Louisiana, USA**
cwenk@tulane.edu

Abstract

It is not known whether the Fréchet distance for general graphs can be efficiently computed. In this paper we consider orientation-preserving isomorphic connected *embedded* graphs and provide a polynomial-time algorithm for computing their Fréchet distance.

1 Introduction

Comparing embedded graphs has important applications and has recently gained attention for comparing reconstructed road networks [1, 2]. See [1, 5] for an overview on different distance measures for comparing planar embedded graphs. The Fréchet distance is a popular distance measure that takes continuity of the shapes into account, and can be computed in polynomial time for curves [3]. While Buchin et al. [4] have shown that Fréchet distance is NP-hard to compute for two polygonal domains, they posed an open problem whether the Fréchet distance between two plane graphs can be computed in polynomial time. In this paper we answer this question in the affirmative by giving a polynomial-time algorithm based on enumerating all orientation-preserving graph isomorphisms.

Plane graphs and isomorphisms. A plane graph is a graph $G = (V_G, E_G)$ together with a planar embedding of the vertices and edges, which maps vertices to point and edges to curves in the plane, such that no two edges cross. For each $v \in V$ let $\rho_G(v)$ be the cyclic ordering of edges incident to v in counter-clockwise order, and $\rho_G^{-1}(v)$ in clockwise order.

Two plane graphs G and H are *planar isomorphic*, $G \cong H$, if there exists a mapping $f\colon V_G \to V_H$ such that (1) f is a bijection, and for all $u, v \in V_G$, (2) $(u, v) \in E_G$ iff $(f(u), f(v)) \in E_H$, and (3) either $f(\rho_G(u)) = \rho_H(f(u))$ or $f(\rho_G(u)) = \rho_H^{-1}(f(u))$. In other words, a planar isomorphism is an isomorphism that preserves the ordering of edges around each vertex. If f maps ρ_G to ρ_H, then f is an *orientation-preserving isomorphism* of G and H. If f maps ρ_G to ρ_H^{-1}, it is an orientation-reversing isomorphism.

Fréchet distance. The Fréchet distance between homeomorphic surfaces P and Q on $\mathbb{R}^2$ is defined as $\delta_F(P, Q) = \inf_{\sigma\colon P \to Q} \sup_{x \in P} d(x, \sigma(x))$, where σ ranges over all orientation-preserving homeomorphisms between P and Q and $d(x, y)$ is an underlying metric, normally the Euclidean distance. This definition applies to two homeomorphic graphs as well. It is well known that if two graphs are homeomorphic, from a topological point of view they are isomorphic. Since the computation of Fréchet distance requires a bijection between the points of two graphs, eventually we need a set of bijective edge mappings induced by these isomorphisms. Thus the Fréchet distance between two graphs can be defined as $\delta_F(G, H) = \min_\alpha \max_{e \in E_G} \delta_F(e, \alpha(e))$, where α ranges over all edge mappings of all

* We thank Maike Buchin, Erin Chambers, Brittany Terese Fasy, Ellen Gasparovic, and Elizabeth Munch for fruitful discussions.
† Partially supported by NSF grant CCF 1637576.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

isomorphisms between G and H. See also [6]. Note that if G and H are not simple, each vertex isomorphism may have more than one different edge mappings since it primarily deals with how vertices connect to each other, but not how vertices and edges are embedded in the plane. For two trees, the Fréchet distance can be computed in polynomial time [4] and for graphs of bounded tree width it is fixed-parameter tractable [6].

2 Polynomial-Time Algorithm

In this section, we present an algorithm for computing the Fréchet distance between two connected plane graphs according to their planar isomorphisms, particularly the orientation-preserving isomorphisms that give some information how planar graphs are embedded in the plane. We give an algorithm with polynomial time for this case. Let $G = (V_G, E_G)$ and $H = (V_H, E_H)$ be two connected plane graphs with piecewise linear embeddings, i.e., edges are embedded by piecewise linear curves. We assume that vertices in V_G and V_H do not have degree two. However, degree-two vertices may be present on the embeddings of the edges, and we allow multiple edges (with different embeddings) between the same vertices as well as loops. We assume $|V_G| = |V_H| = n$ and $|E_G| = |E_H| = m$, and we define N_G (resp., N_H) to be the total complexity of the embedded edges in E_G (resp., E_H). In the algorithm, we only consider the orientation-preserving isomorphisms of two graphs since the computation for the orientation-reversing isomorphisms is similar. One of the main tasks is to find all edge mappings induced by the orientation-preserving isomorphisms between G and H. We assume that G and H in the topological sense are simple graphs. If G and H are simple, it is obvious that each isomorphism has a unique edge mapping. At the end of this section, we discuss how to handle the case that two graphs are non-simple.

Before finding all orientation-preserving isomorphisms between G and H, we show that there are at most $2m$ orientation-preserving isomorphisms. Weinberg [8] indicated that the number of automorphisms of a triconnected plane graph is at most $4m$, half from the orientation-preserving and half from the orientation-reversing. We prove that $2m$ is an upper bound of the number of orientation-preserving isomorphisms for general plane graphs.

▶ **Lemma 2.1.** *Let G and H be orientation-preserving isomorphic. For any $(u_1, u_2) \in E_G$ and any $v_1, v_2 \in V_H$, if there is an orientation-preserving isomorphism $f\colon V_G \to V_H$ such that $f(u_1) = v_1$ and $f(u_2) = v_2$, then f is unique.*

Proof. Since G and H are orientation-preserving isomorphic, there exists an orientation-preserving isomorphism f that maps u_1 and u_2 to v_1 and v_2 in V_H, respectively. $(u_1, u_2) \in E_G$ implies that $(f(u_1), f(u_2)) = (v_1, v_2) \in E_H$.

Let $\{u_1, u_2, ..., u_i, ..., u_n\}$ be the set of all vertices in V_G that is listed as the order of being visited by an arbitrary graph traversal technique starting from u_1 then u_2. Assume that there are two orientation-preserving isomorphisms $f, g\colon V_G \to V_H$ such that $f(u_1) = g(u_1) = v_1$ and $f(u_2) = g(u_2) = v_2$. We show that $f = g$. Let $\phi_v\colon \rho(v) \times \{1, \ldots, |\rho(v)|\} \to \rho(v)$ be a function such that $\phi_v(u, m)$ is the element obtained by walking around v in a clockwise direction for m steps starting at $u \in \rho(v)$. Suppose that $f(u_i) = g(u_i)$ for any $1 \le i \le k$, where $k \ge 2$. For u_{k+1}, since this set is ordered by the traversal technique, there exists a vertex u_x such that $x \le k$ and $(u_x, u_{k+1}) \in E_H$. Consider the following cases:

Case 1: If $u_x = u_1$, then both u_2 and u_{k+1} are neighbors of u_1. Let m_1 be the number such that $\phi_{u_1}(u_2, m_1) = u_{k+1}$. We know that $\rho(f(u_1)) = \rho(v_1) = f(\rho(u_1))$ and $\rho(g(u_1)) = \rho(v_1) = g(\rho(u_1))$. Hence $f(u_{k+1}) = g(u_{k+1}) = \phi_{f(u_1)}(f(u_2), m_1)$.

Case 2: If $u_x \neq u_1$, there exists a vertex u_y such that $(u_y, u_x) \in E_H$ and $1 \le y < x$.

By the hypothesis, $f(u_y) = g(u_y)$ and $f(u_x) = g(u_x)$. Let m_2 be the number such that $\phi_{u_x}(u_y, m_2) = u_{k+1}$. Then $f(u_{k+1}) = g(u_{k+1}) = \phi_{f(u_x)}(f(u_y), m_2)$. ◀

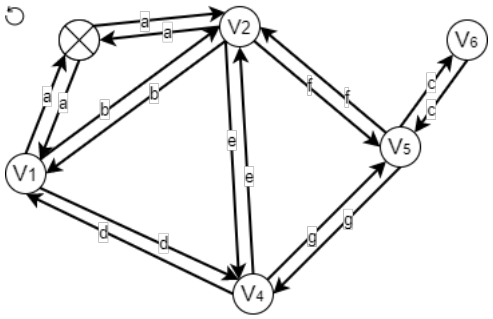


Figure 1 In the graph G above, $\rho_G(v_2) = (v_4, v_5, v_{1a}, v_{1b})$ and $code(G, v_5, v_2) = 012303132121040$.

▶ **Theorem 2.2.** *There are at most $2m$ orientation-preserving isomorphisms between two connected plane graphs.*

Proof. The ordered pair of adjacent vertices $u_1, u_2 \in V_G$ must map to an ordered pair of adjacent vertices $v_1, v_2 \in V_H$. The number of ordered pairs of adjacent vertices is 2 times the number of edges in the graph H. Based on Lemma 2.1, any such mapping c has a unique orientation-preserving isomorphism. Thus, there are at most $2m$. ◀

Weinberg [9] developed a simple $\mathcal{O}(n^2)$ algorithm for testing isomorphism of 3-connected planar graphs. This algorithm mainly depends on the fact that a 3-connected planar graph only has two unique embeddings [10], namely two unique orders of edges around each vertex. There is also a linear time algorithm for this problem [7]. Note that we compute the Fréchet distance by enumerating all possible orientation-preserving isomorphisms and this requires $\mathcal{O}(n^2)$ by Theorem 2.2. We adopt Weinberg's concise algorithm to find all orientation-preserving isomorphisms of plane graphs as well as the induced edge mappings.

The main part of the algorithm can be considered as a DFS-like traversal. Given an undirected plane graph G, we convert it into a directed graph by replacing each undirected edge with two opposite directed edges. We slightly abuse notation to let E_G denote the set of undirected or directed edges. Then the plane graph has an Eulerian cycle since every vertex has an even degree after this modification. We generate a unique Eulerian cycle with respect to ρ with starting edge (s, t) as follows:

1. When an unvisited vertex v is reached on a directed edge (u, v), visit the vertex on the edge succeeding (u, v) in $\rho(v)$. If (u, v) has no next edge in $\rho(v)$ (e.g. $\deg(v) = 1$), visit u again on the directed edge (v, u).
2. When a visited vertex v is reached on a directed edge (u, v), if the opposite directed edge (v, u) is unused, then visit u again on the directed edge (v, u).
3. When a visited vertex v is reached on a directed edge (u, v), if the opposite directed edge (v, u) is used, then visit the vertex on the next unused edge of (u, v) in $\rho(v)$.

During the procedure we relabel vertices by starting with 0 and increasing the label by 1 for each unvisited vertex. We denote the label of v as $N(v)$. We now construct a sequence of numbers to represent the Eulerian cycle by displaying the labels of vertices in order of being reached every time, see Figure 1. We denote this sequence of natural numbers as $code(G, s, t)$,

where s is the starting vertex and t is the next vertex to visit. Obviously, in $code(G, s, t)$, the numbers 0 and 1 are always the first two integers that come from $N(s)$ and $N(t)$. The length of $code(G, s, t)$ is $2m + 1$ since it traverses each directed edge once and eventually returns to the starting vertex s. Weinberg claims that two 3-connected planar graphs G and H are isomorphic if and only if $code(G, s, t) = code(H, s', t')$ for some directed $(s, t) \in E_G$ and directed $(s', t') \in E_H$. However, this is not true if they are general plane graphs. Given two undirected plane graphs G and H, we check and find all orientation-preserving isomorphisms by the following method:

Choose any ordered pair of adjacent vertices $s, t \in V_G$ and construct $code(G, s, t)$. For each ordered pair of adjacent vertices $s', t' \in V_H$, if $code(G, s, t) = code(H, s', t')$, define the function $f : V_G \to V_H$ such that $f(v) = u$ if $N(v) = N(u)$. Then if $f(\rho_G(v)) = \rho_H(f(v))$ for all $v \in V_G$, f is an orientation-preserving isomorphism of G and H. Otherwise, they are not orientation-preserving isomorphic.

▶ **Theorem 2.3.** *For an ordered pair of adjacent vertices $s, t \in V_G$, the graphs G and H are orientation-preserving isomorphic if and only if there exists an ordered pair of adjacent vertices $s', t' \in V_H$ such that $code(G, s, t) = code(H, s', t')$ and $f(\rho_G(v)) = \rho_H(f(v))$ for all $v \in V_G$, where the function $f : V_G \to V_H$ is defined by $f(v) = u$ if $N(v) = N(u)$.*

Proof. In the code formed by the Eulerian cycle, every vertex v has a distinct natural number $N(v)$. Then $code(G, s, t) = code(H, s', t')$ implies G and H have the same number of vertices and every vertex $v \in V_G$ corresponds to exactly one vertex $u \in V_H$. Hence f is a bijection. In the $code(G, s, t) = code(H, s', t') = (01...c_i...0)$, each pair of neighboring numbers represents a directed edge in the graphs. For any ordered $(x, y) \in E_G$, there exists (c_i, c_{i+1}) in the code such that $N(x) = c_i$ and $N(y) = c_{i+1}$. Since $code(H, s', t')$ also contains the pair (c_i, c_{i+1}), there exists an ordered $(x', y') \in E_H$ such that $N(x') = c_i$ and $N(y') = c_{i+1}$. By definition of f we have $f(x) = x'$ and $f(y) = y'$, which imply that if $(x, y) \in E_G$, then $(f(x), f(y)) \in E_H$. Conversely, $(f(x), f(y)) \in E_H$ implies $(x, y) \in E_G$. Finally, $f(\rho_G(v)) = \rho_H(f(v))$ for all $v \in V_G$ indicates that f is an orientation-preserving isomorphism.

Suppose that G and H are orientation-preserving isomorphic. There exists an orientation-preserving isomorphism $f : V_G \to V_H$ such that $f(s) = s'$, $f(t) = t'$ and $(s', t') \in E_H$. Since f is a vertex bijection and the ordering of edges of each vertex is preserved, $|E_G| = |E_H|$. Let $\{u_0, u_1, ..., u_{2|E|}\}$ and $\{v_0, v_1, ..., v_{2|E|}\}$ be the Eulerian cycles starting from the ordered pairs (s, t) and (s', t'), where $(u_0, u_1) = (s, t)$ and $(v_0, v_1) = (s', t')$. We have that $f(u_0) = v_0$ and $f(u_1) = v_1$. Assume that $f(u_i) = v_i$ for any $0 \le i \le k$, where $k \ge 1$. Then we have $f(u_{i+1}) = v_{i+1}$. This is because $f(\rho_G(u_i)) = \rho_H(v_i)$ and the used edges when reaching u_i are exactly corresponding to the used edges when reaching v_i with respect to the orientation-preserving isomorphism. Hence $\{v_0, v_1, ..., v_{2|E|}\} = \{f(u_0), f(u_1), ..., f(u_{2|E|})\}$. By the rules of assigning a vertex a natural number, $code(G, s, t) = code(H, s', t')$. ◀

If G and H are simple, it is immediately clear that an orientation-preserving isomorphism f can induce a unique edge mapping $g : E_G \to E_H$ by $g((u, v)) = (f(u), f(v))$. When G and H are non-simple, i.e., contain multiple edges or loops, we can acquire all edge mappings from their codes. This is because a code travels all edges, and any adjacent pair in the code represents a directed edge. In this situation, the multiple edges between two vertices need to be properly distinguished and a vertex may need two incident edges to indicate each of its loops. The exact edge mapping can be obtained by tracing each pair of adjacent numbers of their codes in parallel. Once we have all edge mappings between G and H, finally, the Fréchet distance between G and H can be computed by directly following the formula. For the Fréchet distance between two polygonal edges e_1 and e_2, we adopt the algorithm with

$\mathcal{O}(N_{e_1}N_{e_2}\log(N_{e_1}N_{e_2}))$ time in [3], where N_{e_1} and N_{e_2} are the complexity of e_1 and e_2, respectively.

▶ **Theorem 2.4.** *The Fréchet distance between two orientation-preserving isomorphic connected plane graphs can be computed in $\mathcal{O}(N_G N_H \log^2(N_G N_H))$ time.*

Proof. For each edge mapping g induced by an orientation-preserving isomorphism, we compute the Fréchet distance $\delta_F((u,v), g(u,v))$ for all edges $(u,v) \in E_G$. This requires the time for computing the Fréchet distance between each pair of polygonal edges for m pairs in one edge mapping. By Theorem 2.2, there are $\mathcal{O}(m)$ orientation-preserving isomorphisms. We enumerate all of them in total $\mathcal{O}(m^2)$ time using Theorem 2.2. Hence, in the direct computation, we may need to compute the Fréchet distance between two edges for $\mathcal{O}(m^2)$ pairs. However, the total time for computing the Fréchet distance between each pair of edges is $\mathcal{O}(N_G N_H \log(N_G N_H))$ [6] if both graphs are not closed curves. The last case that has not been covered is when a connected plane graph only has vertices of degree 2, i.e., the graph is a closed curve, since we assume that there are no vertices of degree 2 in computing orientation-preserving isomorphisms. It is known that the Fréchet distance of two closed curves can be computed in $\mathcal{O}(N_G N_H \log^2(N_G N_H))$ time [3]. ◀

3 Conclusion

When the orientation of edges is considered, the number of possible edge mappings is significantly reduced. We conclude that the Fréchet distance between two orientation-preserving isomorphic graphs can be computed in polynomial time.

References

1. M. Ahmed, S. Karagiorgou, D. Pfoser, and C. Wenk. *Map Construction Algorithms*. Springer International Publishing, first edition, 2015. doi:10.1007/978-3-319-25166-0.
2. Mahmuda Ahmed, Sophia Karagiorgou, Dieter Pfoser, and Carola Wenk. A comparison and evaluation of map construction algorithms. *Geoinformatica*, 19(3):601–632, 2015.
3. Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *Internat. J. Comput. Geom. Appl.*, 5(1-2):75–91, 1995.
4. Kevin Buchin, Maike Buchin, and André Schulz. Fréchet distance of surfaces: Some simple hard cases. In *European Symposium on Algorithms*, pages 63–74. Springer, 2010.
5. M. Buchin, E. Chambers, P. Fang, B.T. Fasy, E. Gasparovic, E. Munch, and C. Wenk. Distance measures for embedded and immersed graphs, 2021. In preparation.
6. Maike Buchin, Amer Krivosija, and Alexander Neuhaus. Computing the Fréchet distance of trees and graphs of bounded tree width. In *Proc. 36th European Workshop on Computational Geometry (EuroCG)*, 2020.
7. John E Hopcroft and Jin-Kue Wong. Linear time algorithm for isomorphism of planar graphs. In *Proceedings of the sixth annual ACM symposium on Theory of Computing*, pages 172–184, 1974.
8. Louis Weinberg. On the maximum order of the automorphism group of a planar triply connected graph. *SIAM Journal on Applied Mathematics*, 14:729–738, 1966.
9. Louis Weinberg. A simple and efficient algorithm for determining isomorphism of planar triply connected graphs. *IEEE Transactions on Circuit Theory*, 13:142–148, 1966. doi:10.1109/TCT.1966.1082573.
10. Hassler Whitney. A set of topological invariants for graphs. *American Journal of Mathematics*, 55:231–235, 1933.

Sampling Hyperplanes and Revealing Disks*

Haim Kaplan[1], Alexander Kauer[†2], Wolfgang Mulzer[‡2], and Liam Roditty[3]

1 **School of Computer Science, Tel Aviv University**
haimk@tau.ac.il
2 **Institut für Informatik, Freie Universität Berlin, Berlin, Germany**
[akauer, mulzer]@inf.fu-berlin.de
3 **Department of Computer Science, Bar Ilan University**
liamr@macs.biu.ac.il

Abstract

We present a data structure which allows detecting when disks of a set B are no longer intersected by disks of set of disk A when deleting its disks. Preprocessing A, B and deleting n disks of A with detecting all newly revealed disks of B requires $\mathcal{O}(|B|\log^4|A| + |A|\log^5(|A|)\lambda_6(\log|A|) + n\log^9(|A|)\lambda_6(\log|A|))$ expected time, where $\lambda_6(\cdot)$ is the Davenport-Schinzel bound of order 6.

To construct this data structure, we extend known dynamic lower envelope data structures for hyperplanes and bivariate functions of constant description complexity with a linear size lower envelope in $\mathbb{R}^3$, such that they allow sampling of a random element not above a given point in $\mathcal{O}(\log^3 n)$ expected time.

1 From Graph Connectivity to Disk Sampling

Graph connectivity plays a fundamental role in algorithms and data structures. The dynamic variant where edges can be inserted or deleted is reasonably well understood [6–8,11,14], with data structures supporting updates and queries determining the connectivity of two vertices in polylogarithmic time. Updating vertices seems significantly harder, as a single update can have a large impact on the overall structure. Chan et al. [4, Theorem 1] presented a data structure allowing vertex updates in $\widetilde{\mathcal{O}}(m^{2/3})$ amortized time and queries in $\widetilde{\mathcal{O}}(m^{1/3})$ time, where m is the number of possible edges of the graph that need to be known in advance.

The vertices of geometric intersection graphs correspond to geometric objects and its edges to intersections. As they restrict possible graphs, faster solutions may be within reach. However, work is required to find edges affected by an update. Chan et al. [4, Theorem 5] gave a general method for various objects with sub-linear update and query times.

In this work we present a data structure, which allows detecting when a disk is no longer intersected by any disk of a set after deletions. We call such a disk *revealed*. This data structure can be used as a component in maintaining connectivity information in deletion-only disk intersection graphs [9].

To construct it, we first describe data structures for randomly sampling a hyperplane or a continuous function of constant description complexity not above a given point in $\mathbb{R}^3$, which might be of interest of its own.

* Supported in part by grant 1367/2016 from the German-Israeli Science Foundation (GIF).
† Supported in part by grant 1367/2016 from the German-Israeli Science Foundation (GIF) and by the German Research Foundation within the collaborative DACH project *Arrangements and Drawings* as DFG Project MU 3501/3-1.
‡ Supported in part by ERC StG 757609.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

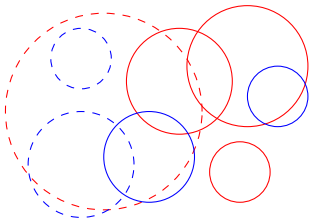


Figure 1 When removing a red disk, we want to obtain all blue disks intersecting this red disk but no other red disk. After removing the dashed red disk, the dashed blue disks need to be obtained.

2 Preliminaries

Let A, B be two sets of disks in $\mathbb{R}^3$, such that for each disk $b \in B$ exists at least one disk $a \in A$ it intersects. We now want to remove a disk $a \in A$ and obtain all disks $b \in B$ which no longer have any disk from A they intersect. We call such disks *revealed*. See Figure 1.

A data structure for detecting such revealed disks is constructed over the course of Sections 4 and 5, which concludes in Theorem 5.3. Its central idea is representing intersections sparsely via assigning each $b \in B$ repeatedly to one random $a \in A$ it intersects until there is none left and b is revealed.

Obtaining such a random disk of a given semi-dynamic set which intersects a query disk is the main problem. We will build upon the dynamic lower envelope data structures by Kaplan et al. Those maintain a dynamic set of hyperplanes [10, Section 7] or continuous bivariate functions of constant description complexity [10, Section 8] in $\mathbb{R}^3$ under insertions and deletions. Also, they support *vertical ray shooting queries* into their lower envelopes. The second variant is an extension of the first, and the first is a slight extension of a data structure by Chan [3]. We will briefly describe the first variant in Section 3.

The data structures use vertical k-shallow $(1/r)$-cuttings as their most integral part. Let $\mathcal{A}(H)$ be the arrangement of hyperplanes H in $\mathbb{R}^3$. The *k-level* L_k of $\mathcal{A}(H)$ is the closure of all points of $\bigcup H$ with k hyperplanes of H strictly below. Then, $L_{\leq k}$ is the union of the levels until L_k. A *vertical k-shallow $(1/r)$-cutting* is a set Λ of pairwise openly disjoint prisms, such that the union of Λ covers $L_{\leq k}$, the interior of each $\tau \in \Lambda$ is intersected by at most $|H|/r$ hyperplanes of H, and each prism is vertical (i.e. it consists of a triangle and all points below it). Some or all points of a prism's top may lie at infinity. The number of prisms is the *size* of the cutting. Using the algorithm by Chan and Tsakalidis a vertical $\Theta(|H|/r)$-shallow $1/r$-cutting of size $\mathcal{O}(r)$ can be created in $O(|H| \log r)$ time [5]. The *conflict list* $\mathrm{CL}(\tau)$ of a prism τ is the set of all hyperplanes crossing its interior.

This notion can also be extended to bivariate functions. Then the regions are vertical pseudo-prisms, where the top is limited by a pseudo-trapezoid part of a function [10, Section 3].

3 Vertical Ray Shooting Queries for Hyperplanes

First, we will construct a data structure for the simpler problem of sampling a random disk containing a given point from a dynamic set of disks. Using linearization [1, 15] we can transform this problem in $\mathbb{R}^2$ into the problem of sampling a hyperplane not above a given point in $\mathbb{R}^3$. This allows us to build upon the data structure for hyperplanes by Kaplan et al.

Their data structure consists of $\mathcal{O}(\log n)$ static substructures of exponentially decaying size, where n is the current number of hyperplanes. Substructures are periodically rebuilt similar to the Bentley-Saxe technique [2] and the whole data structure after $\Theta(n)$ updates. Each

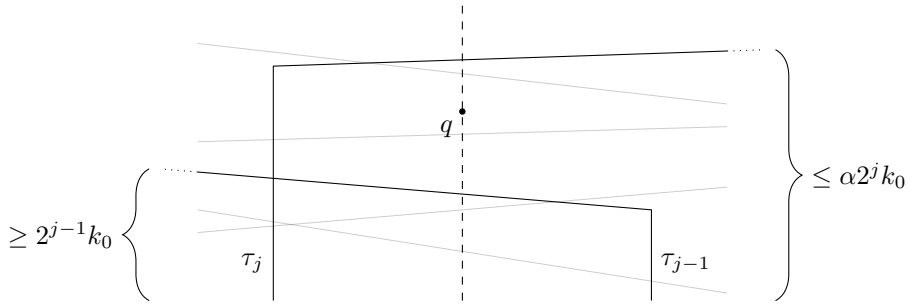


Figure 2 The situation after walking up for q and the number of hyperplanes of $\mathrm{CL}(\tau_j)$ intersecting.

substructure of n' elements consists of a hierarchy of vertical shallow cuttings $\{\Lambda_j\}_{0\leq j\leq m+1}$ with $m \in \mathcal{O}(\log n')$. Let $k_j = 2^j k_0$, n_j be the number of planes in Λ_{j+1}, and k_0, α be constants. Λ_{m+1} consists of a single prism covering all of $\mathbb{R}^3$. Λ_j for $j < m+1$ consists of a vertical k_j-shallow $(\alpha k_j/n_j)$-cutting of the hyperplanes of Λ_{j+1}, with hyperplanes intersecting too many prisms removed. Thus, the prisms of Λ_j cover $L_{\leq k_j}$ of the planes of Λ_{j+1} and each conflict list contains at most αk_j hyperplanes. The removed hyperplanes are reinserted into other substructures, such that each hyperplane is in a substructure's Λ_0. A substructure requires $\mathcal{O}(n' \log n')$ space and building it requires $\mathcal{O}(n' \log^2 n')$ time.

Vertical ray shooting queries are done via searching the Λ_0 of all $\mathcal{O}(\log n)$ substructures for the prism intersecting the vertical line containing the given position in $\mathcal{O}(\log n)$ time each. Then, all obtained prisms are searched for the lowest plane along the line in $\mathcal{O}(\log n)$ time.

Insertions are handled with a Bentley-Saxe approach via rebuilding multiple smaller substructures into new substructures periodically. They require $\mathcal{O}(\log^3 n)$ amortized time.

Deletions are handled differently. In all substructures hyperplanes are only marked as deleted and ignored in queries. As described above, queries per substructure are done in their Λ_0 only. Thus, the lowest non-deleted hyperplane along a given vertical line might not be contained in the prism of Λ_0 intersecting said line. It may require the corresponding prism of Λ_1 or an even higher Λ_j, as prisms from higher cuttings can intersect more hyperplanes.

To handle this, when deleting a hyperplane in a substructure all prisms containing it are identified. For each such prism an individual counter is incremented. If it reaches a fraction of $f = \frac{1}{2\alpha}$ of the size of its conflict list, the prism and its hyperplanes are marked as purged in the substructure. When hyperplanes are first marked as purged, they are also reinserted into the data structure anew. Hyperplanes marked as purged are skipped in queries as well.

Consider prisms $\tau_1, \ldots, \tau_{m+1}$ from $\Lambda_1, \ldots, \Lambda_{m+1}$ intersecting a vertical line. The idea behind the purging is that $|\mathrm{CL}(\tau_j)| \leq \alpha k_j$ and Λ_{j-1} covers $L_{\leq k_{j-1}}$ of the hyperplanes of Λ_j, see Figure 2. Thus, a fraction of at least $\frac{k_{j-1}}{\alpha k_j} = f$ of $\mathrm{CL}(\tau_j)$ is intersected by τ_{j-1}. A plane first appearing in $\mathrm{CL}(\tau_j)$ then can only be the lowest along the vertical line if all from $\mathrm{CL}(\tau_{j-1})$ have been deleted and thus a fraction of f of $\mathrm{CL}(\tau_j)$ as well. Then, τ_j would have been purged and its hyperplanes reinserted into other substructures. Also, each hyperplane is contained in at most one substructure in Λ_0 without being marked as purged or deleted.

Deletions require $\mathcal{O}(\log^5 n)$ amortized time and overall $\mathcal{O}(n \log n)$ space is needed.

4 Sampling Hyperplanes

When at least a fraction of all hyperplanes in $\mathrm{CL}(\tau_j)$ is intersected by any τ_{j-1} and is not marked as deleted, we can sample inside $\mathrm{CL}(\tau_j)$ to get a non-deleted hyperplane intersecting

τ_{j-1} with a minimum probability. For this, we lower the purging threshold to $f' = \frac{1}{4\alpha}$.

▶ **Lemma 4.1.** *The Kaplan et al. data structure for hyperplanes [10, Section 7] with $f' = \frac{1}{4\alpha}$ as purging threshold works as desired with the same asymptotic time and space bounds.*

The correctness of the argument in Section 3 (and the proof by Kaplan et al. [10, Lemma 7.6]) is unchanged, as prisms are just purged earlier. The run time analysis [10, Lemma 7.7] requires an adjustment of constants only[1], and the asymptotic space bound is unaffected as well.

▶ **Lemma 4.2.** *Given prisms τ_j of Λ_j and τ_{j-1} of Λ_{j-1} with $j \geq 1$ from the same substructure and intersected by the same vertical line. If τ_j not has been purged with threshold $f' = \frac{1}{4\alpha}$, at least $\frac{1}{4\alpha}|CL(\tau_j)|$ of its hyperplanes are intersected by τ_{j-1} and are not marked as deleted.*

Proof. The prism τ_{j-1} intersects at least $k_{j-1} = 2^{j-1}k_0$ hyperplanes of Λ_j in its interior, as τ_{j-1} is from a vertical k_{j-1}-shallow cutting of the hyperplanes of Λ_j. Due to the vertical line these hyperplanes must all be contained in $\mathrm{CL}(\tau_j)$ as well. See Figure 2. Also, the prism τ_j intersects at most $\alpha k_j = \alpha 2^j k_0$ hyperplanes, as it is built from a $(\alpha k_j/n_j)$-cutting.

Thus, if τ_j has not been purged due to deletions, it must contain a fraction of

$$> \frac{2^{j-1}k_0 - \frac{1}{4\alpha} \cdot \alpha 2^j k_0}{\alpha 2^j k_0} = \frac{2^{j-2}k_0}{\alpha 2^j k_0} = \frac{1}{4\alpha} \tag{1}$$

non-deleted hyperplanes intersecting τ_{j-1}. ◀

Kaplan et al. [10] already observed the following. For each individual substructure, the ceilings of the prisms of Λ_j form a polyhedral terrain $\overline{\Lambda}_j$. Due to the removal of planes between steps during creation the terrain $\overline{\Lambda}_j$ does not necessarily lie below $\overline{\Lambda}_{j+1}$. Nevertheless, the number of hyperplanes in Λ_j below any point is less or equal than the number in Λ_{j+1}. This is in particular valid for those points in or above $\overline{\Lambda}_j$.

To sample a hyperplane not above a given point we walk the Λ_j from $j = 0$ upwards. At each step we locate the vertical prism which contains the point or has it above in $\mathcal{O}(\log n)$ time, as it is done in Λ_0 in the original data structure. In case the point is located *inside* the prism we stop, otherwise we continue. This allows us to apply Lemma 4.2. See Figure 2.

▶ **Theorem 4.3.** *The lower envelope of hyperplanes in $\mathbb{R}^3$ can be maintained dynamically, where each insertion takes $\mathcal{O}(\log^3 n)$ amortized time, each deletion takes $\mathcal{O}(\log^5 n)$ amortized time, vertical ray shooting queries take $\mathcal{O}(\log^2 n)$ time, and sampling a random hyperplane not above a given point takes $\mathcal{O}(\log^3 n)$ expected time, where n is the number of hyperplanes when the operation is performed. The data structure requires $\mathcal{O}(n \log n)$ storage.*

Proof. We construct the data structure for hyperplanes by Kaplan et al. [10, Section 7] with $f' = \frac{1}{4\alpha}$ as the purging threshold and without their memory optimization (which we omitted in Section 3). According to Lemma 4.1 the correctness and asymptotic bounds are unchanged. We construct for all substructures for all Λ_j, $j \geq 1$ point location data structures. This requires $\mathcal{O}(n \log n)$ storage and the run time can be subsumed in the respective creation of the vertical shallow cutting.

Sampling a hyperplane not above a query point q then can be done as follows. For each substructure the first Λ_j and corresponding τ_j are obtained where q is inside τ_j, as described

[1] In the original proof b' has to be chosen larger (e.g. $b' \geq 8\alpha b''$), as purging a prism τ releases $\geq (\frac{b'}{4\alpha} - b'')|\mathrm{CL}(\tau)| \log N$ credits now.

above. In case the prism was purged or the prism is τ_0 and no non-deleted hyperplane lies not above q, this substructure is skipped. This required $\mathcal{O}(\log^3 n)$ time altogether.

All hyperplanes of a substructure's Λ_0 not above q are contained in $\mathrm{CL}(\tau_j)$. Hence, if all substructures were skipped there is no hyperplane not above q. Otherwise, hyperplanes are sampled from all prisms obtained simultaneously, until the result is not marked as deleted, is not marked as purged in its substructure, and is contained in the substructure's Λ_0 (i.e. was not removed during creation).

If we stopped at $j \geq 1$, the top of τ_{j-1} lies not above q, and we can apply Lemma 4.2. Thus, each non-skipped conflict list contains a fraction of at least $\min(f', 1/k_0)$ non-deleted hyperplanes not above q. Recall that each non-deleted hyperplane is contained in exactly one substructure in Λ_0 without being marked as purged. Hence, each non-deleted hyperplane is sampled with equal probability and each sampling returns with a probability of at least $\min(f', 1/k_0) \cdot 1/\mathcal{O}(\log n)$ a valid hyperplane. Thus, we expect $\mathcal{O}(\log n)$ samplings. ◀

▶ **Corollary 4.4.** *Sampling a random disk in $\mathbb{R}^2$ containing a given point from a dynamic set can be implemented with the bounds of Theorem 4.3.*

5 Sampling Disks

Sampling a random disk intersecting a given disk requires another approach, as linearization results in hyperplanes in $\mathbb{R}^4$. Instead, we sample via the distance functions the disks imply.

In addition to the hyperplane lower envelope data structure, Kaplan et al. describe how to create shallow cuttings of totally defined continuous functions $\mathbb{R}^2 \to \mathbb{R}$ of constant description complexity [10, Theorem 8.1, Theorem 8.2]. Afterwards, they plug it as a black box into their hyperplane data structure [10, Theorem 8.3]. Its analysis changes in values only. We adjust the purging threshold and sample as before, while keeping bounds and correctness intact. $\lambda_s(n)$ is the maximum length of a Davenport-Schinzel sequence of order s on n symbols [13].

▶ **Theorem 5.1.** *The lower envelope of totally defined continuous bivariate functions of constant description complexity in $\mathbb{R}^3$, where the lower envelope of any subset has linear complexity, can be maintained dynamically, where each insertion takes $\mathcal{O}(\log^5(n)\lambda_s(\log n))$ amortized expected time, each deletion takes $\mathcal{O}(\log^9(n)\lambda_s(\log n))$ amortized expected time, vertical ray shooting queries take $\mathcal{O}(\log^2 n)$ time, and sampling a random function not above a given point takes $\mathcal{O}(\log^3 n)$ expected time, where n is the number of functions when the operation is performed. The data structure requires $\mathcal{O}(n \log^3 n)$ expected space.*

Using this theorem we can finally sample disks intersecting a given disk and construct a data structure for finding newly revealed disks after deletions.

▶ **Corollary 5.2.** *A set of disks can be maintained dynamically and a random disk sampled intersecting a given disk with the bounds of Theorem 5.1 with $s = 6$.*

Proof. Let d be a disk with center c_d and radius r_d. Then we can represent the distance of any point $p \in \mathbb{R}^2$ from this disk as additively weighted Euclidean metric with $\delta(p, d) = |pc_d| - r_d$, where $|\cdot|$ is the Euclidean distance. The distance functions of multiple disks form a lower envelope of linear complexity [12] and have $s = 6$ [10, Section 9]. Hence, we can build the data structure of Theorem 5.1 with $s = 6$ to maintain the distance functions $\delta(p, d)$ of all disks d. A disk intersecting a given disk q then can be found via sampling a random function not above the point $((c_q)_x, (c_q)_y, r_q)^T$, as every function not above has $\delta(c_q, d) = |c_q c_d| - r_d \leq r_q$. ◀

▶ **Theorem 5.3.** *Let A and B be sets of disks in $\mathbb{R}^2$. We can preprocess A and B, such that elements can be inserted into or deleted from B and elements can be deleted from A while detecting all newly revealed disks of B after each operation. Preprocessing A and B and deleting n disks of A with detecting all newly revealed disks of B requires $\mathcal{O}(m \log^4 |A| + |A| \log^5(|A|)\lambda_6(\log |A|) + n \log^9(|A|)\lambda_6(\log |A|))$ expected time and $\mathcal{O}(|A| \log^3 |A| + m)$ expected space, where m is maximum size of B. Updating B requires $\mathcal{O}(\log^3 |A|)$ expected time.*

Proof. We repeatedly assign each $b \in B$ randomly to an $a \in A$ it intersects. New assignments are made both initially and each time the previous assigned $a \in A$ gets deleted, unless b got revealed. Fix the deletion order $a_{|A|}, \dots a_1$. Each $b \in B$ is reassigned after deleting $a_i \in A$ with probability $1/i$, assuming it intersects all a_i. This results in an expected number of $\leq H_{|A|} \in \mathcal{O}(\log(|A|))$ reassignments per $b \in B$.

We manage A with the data structure of Corollary 5.2. Building it and assigning each $b \in B$ to an $a \in A$ it intersects requires $\mathcal{O}(|A| \log^5(|A|)\lambda_6(\log |A|) + |B| \log^3 |A|)$ expected amortized time. Each deletion in A requires $\mathcal{O}(\log^9(|A|)\lambda_6(\log |A|))$ amortized time plus the time for reassignments. These sum up to $\mathcal{O}(m \log^4 |A|)$ expected time. Updating B needs $\mathcal{O}(\log^3 |A|)$ expected time. The space bound follows from Corollary 5.2. ◀

References

1. P. K. Agarwal and J. Matousek. On range searching with semialgebraic sets. *Discrete Comput. Geom.*, 11(4):393–418, 1994. doi:10.1007/BF02574015.
2. Jon Louis Bentley and James B Saxe. Decomposable searching problems 1. static-to-dynamic transformation. *J. Algorithms*, 1:58, 1980.
3. Timothy M. Chan. A dynamic data structure for 3-d convex hulls and 2-d nearest neighbor queries. *J. ACM*, 57(3):16:1–16:15, 2010. doi:10.1145/1706591.1706596.
4. Timothy M. Chan, Mihai Pătraşcu, and Liam Roditty. Dynamic connectivity: Connecting to networks and geometry. *SIAM J. Comput.*, 40(2):333–349, 2011.
5. Timothy M. Chan and Konstantinos Tsakalidis. Optimal deterministic algorithms for 2-d and 3-d shallow cuttings. *Discrete Comput. Geom.*, 56(4):866–881, 2016. doi:10.1007/s00454-016-9784-4.
6. David Eppstein, Giuseppe F. Italiano, Roberto Tamassia, Robert Endre Tarjan, Jeffery Westbrook, and Moti Yung. Maintenance of a minimum spanning forest in a dynamic plane graph. *J. Algorithms*, 13(1):33–54, 1992.
7. Monika Rauch Henzinger and Valerie King. Randomized fully dynamic graph algorithms with polylogarithmic time per operation. *J. ACM*, 46:502–516, 1999.
8. Jacob Holm, Kristian de Lichtenberg, and Mikkel Thorup. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. *J. ACM*, 48(4):723–760, 2001.
9. Haim Kaplan, Katharina Klost, Kristin Knorr, Wolfgang Mulzer, and Liam Roditty. Deletion only dynamic connectivity for disk graphs. In *Proceedings of the 37th European Workshop on Computational Geometry (EuroCG)*, page 58:1 ff., 2021. Extended abstract.
10. Haim Kaplan, Wolfgang Mulzer, Liam Roditty, Paul Seiferth, and Micha Sharir. Dynamic planar Voronoi diagrams for general distance functions and their algorithmic applications. *Discrete Comput. Geom.*, 64(3):838–904, 2020. doi:10.1007/s00454-020-00243-7.
11. Mihai Pătraşcu and Mikkel Thorup. Planning for fast connectivity updates. In *Proc. 48th Annu. IEEE Sympos. Found. Comput. Sci. (FOCS)*, pages 263–271, 2007.
12. Micha Sharir. Intersection and closest-pair problems for a set of planar discs. *SIAM Journal on Computing*, 14(2):448–468, 1985.
13. Micha Sharir and Pankaj K. Agarwal. *Davenport-Schinzel sequences and their geometric applications.* Cambridge University Press, 1995.

14 Mikkel Thorup. Near-optimal fully-dynamic graph connectivity. In *Proc. 32nd Annu. ACM Sympos. Theory Comput. (STOC)*, pages 343–350, 2000.

15 Andrew C. Yao and F. Frances Yao. A general approach to D-dimensional geometric queries. In *Proc. 17th Annu. ACM Sympos. Theory Comput. (STOC)*, pages 163–168, 1985. doi:10.1145/22145.22163.

On Minimum-Complexity Graph Simplification*

Omrit Filtser[1], Majid Mirzanezhad[2], and Carola Wenk[2]

1 **Department of Applied Mathematics and Statistics, State University of New York at Stony Brook, N.Y., U.S.**
omrit.filtser@gmail.com

2 **Department of Computer Science, Tulane University, L.A., U.S.,**
mmirzane@tulane.edu
cwenk@tulane.edu

Abstract

Simplifying graphs is a very applicable problem in computational geometry. Given a graph G and a threshold δ, the minimum-complexity graph simplification problem asks for computing an alternative graph G' of minimum number of vertices and/or edges so that the distance between G and G' is at most δ. In this paper we propose several NP-hardness and algorithmic results depending on the type of input/output graphs and the type of distance measures considered between them.

1 Introduction

Unlike curve simplification, simplifying higher structural input objects such as trees and graphs has not been extensively studied in computational geometry. This problem has many applications in Geographic Information Systems (GIS), image processing, mesh simplification, molecular biology, etc. [3, 4, 7, 10]. There are a few works that set out to study approximating a planar subdivision of a map with the minimum number of links under some topological constraints [5, 8]. Several algorithms applied to GIS data are based on map schematization [12] in which, roughly speaking, the main topological structure of the map remains the same and paths with vertices of degree two become simplified. A generalization of map schematization under topological constraints such as facet preservation, non-intersecting boundary can be found in [6, 11].

Most of the map simplification problems under some topological constraints fall into NP-hard and/or APX-hard classes of problems [5, 8]. There are many different variants of the (graph) simplification induced by the input parameters/constraints such as the type of the distance measure, e.g., Fréchet, Hausdorff distances, the direction (if the distance is not symmetric) and the way of applying it (local vs. global), topological constraints (noncrossing, facet maintenance), etc. Note that some distances are not symmetric, therefore it matters in which direction the distance is applied. Our systematic study provides different variants of the problem as the input and output vary between trees and graphs.

In this paper we study different variants of the following generic problem: Let $\delta > 0$ be a real, $\mathsf{D}(\cdot,\cdot)$ be a distance measure, and $G = (V, E)$ be a graph immersed in $\mathbb{R}^d$, whose edges in E are straight-line segments between the vertices in V. We aim to compute an alternative graph G', with the minimum-complexity (min-edge and/or min-vertex) satisfying $\mathsf{D}(G, G') \leq \delta$. For instance, *a minimum-edge graph-to-tree simplification from output to input* refers to a variant in which a graph is simplified by a tree when the distance is applied from the output tree to the input graph. We consider two Fréchet-like distance measures between graphs; the *graph distance* proposed in [1] and *traversal distance* in [2]. We will

* This project has been supported by NSF grant (AitF: NSF-CCF 1637576)

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

define these distances more formally below and we emphasize that none of these distances are metric. In general, given two graphs G_1 and G_2 as the input, the problem of computing the graph distance from G_1 to G_2 admits polynomial-time algorithm when G_1 is tree and G_2 is a graph [1]. The problem becomes NP-hard when G_1 and G_2 are arbitrary graphs. Alt et al. in [2] proposed an $O(nm \log^2 nm)$ time algorithm to compute the traversal distance, from G_1 to G_2, where n and m are the number of vertices of the two graphs.

Notations. Let P be a polygonal curve. We treat P as a continuous map $P : [1, n] \to \mathbb{R}^d$, where the i-th edge is linearly parameterized as $P(i + \lambda) = (1 - \lambda)p_i + \lambda p_{i+1}$ with the i-th vertex p_i and $\lambda \in [0, 1]$. We write $P[s, t]$ for the subcurve between $P(s)$ and $P(t)$ and $\langle P(s)P(t)\rangle$ for the link connecting them. Let G be an immersed graph in $\mathbb{R}^d$. We denote the set of vertices of G by $V(G)$ and the set of edges by $E(G)$. A continuous mapping $f : [0, 1] \to G$ is called a *traversal* of G if it is surjective, and a *partial traversal* of G if it is not necessarily surjective. Given two curves $P : [1, n] \to \mathbb{R}^d$ and $Q : [1, m] \to \mathbb{R}^d$, the Fréchet distance between P and Q is defined as: $\mathsf{F}(P, Q) = \inf_{f,g} \max_{t\in[0,1]} \|P(f(t)) - Q(g(t))\|$, where $f : [0, 1] \to [0, n]$ and $g : [0, 1] \to [0, m]$ are continuous, surjective, and non-decreasing functions. In the definition above if f and g are not required to be non-decreasing functions, then the obtained distance is called *the weak Fréchet distance* denoted by $\mathsf{wF}(P, Q)$.

Given two graphs G_1, G_2 immersed in $\mathbb{R}^d$, their *traversal distance* [2] is: $\overrightarrow{\delta_{\mathsf{T}}}(G_1, G_2) = \inf_{f,g} \max_{t\in[0,1]} \|f(t) - g(t)\|$, where $f : [0, 1] \to G_1$ is a traversal of G_1, that is a continuous surjective function, and $g : [0, 1] \to G_2$ is a partial traversal of G_2 that is a continuous function only. A *graph mapping* is a function $s : G_1 \to G_2$ that (1) maps each vertex $v \in V(G_1)$ to a point $s(v)$ on an edge of G_2, and (2) maps each edge $\langle uv\rangle \in E(G_1)$ to a simple path from $s(u)$ to $s(v)$ in the embedding of G_2. The *directed (strong) graph distance* between G_1 and G_2 is: $\overrightarrow{\delta_G}(G_1, G_2) = \inf_{s:G_1\to G_2} \max_{e\in E(G_1)} \mathsf{F}(e, s(e))$, and the *(undirected) graph distance* between G_1 and G_2 is $\delta_G(G_1, G_2) = \max(\overrightarrow{\delta_G}(G_1, G_2), \overrightarrow{\delta_G}(G_2, G_1))$. Correspondingly, the *directed (weak) graph distance* denoted by $\overrightarrow{\delta_{wG}}$ is obtained by replacing the F with wF in the definition above.

Our results: We show that the min-edge graph-to-graph simplification under the graph distances from input to output is NP-hard (Thm. 2.2). We show that this problem for input and output trees is NP-hard when one has to map the leaves of the input to the output under the graph and traversal distances (Thm. 3.1). If we flip the direction of the graph distance in the latter variant, one can give an $O(kn^5)$ time algorithm (Thm. 4.2) where k is the number of leaves of the of the input tree mapped from the output under the graph distance.

2 Minimum-Edge Graph-Graph Simplification

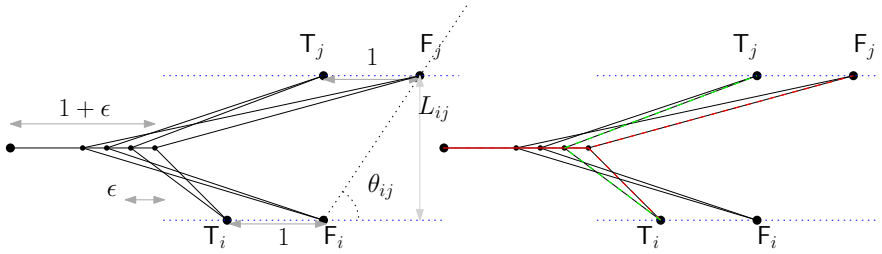
We prove that computing the minimum-edge graph-to-graph simplification under graph and traversal distances from input to output is NP-hard. Note that in this section the simplified graph is a **subgraph** of the input and it selects its vertices from the vertices of the input. We reduce from a specific variant of Max-2SAT, in which the variable graph $G_{\mathcal{F}}$ is a bipartite graph, where $\mathcal{F}$ is a formula on the set of variables $X = \{X_1, X_2, \cdots, X_n\}$, and the set of clauses is $\mathcal{C}$. We call this problem Bipartite-Max-2SAT. In the variable graph, nodes are the variables in X and two nodes are connected if both belong to the same clause in $\mathcal{C}$.

▶ **Lemma 2.1.** Bipartite-Max-2SAT *is NP-hard.*

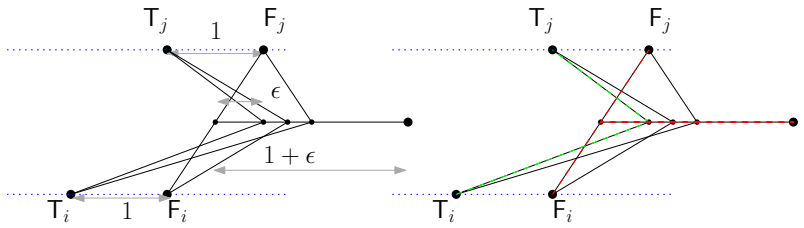
Proof Sketch. Replacing every clause $C = (x \vee y \vee z)$ in the 3-SAT formula with the following 16 clauses results in a bipartite variable graph: $\mathcal{F}_C = (\bar{x} \vee a) \wedge (\bar{x} \vee \bar{b}) \wedge (\bar{y} \vee b) \wedge (\bar{y} \vee \bar{c}) \wedge (\bar{z} \vee$

$c) \wedge (\bar{z} \vee \bar{a}) \wedge (\bar{w} \vee d_1) \wedge (\bar{w} \vee d_2) \wedge (\bar{w} \vee d_3) \wedge (x \vee \bar{d_1}) \wedge (y \vee \bar{d_2}) \wedge (z \vee \bar{d_3}) \wedge (x) \wedge (y) \wedge (z) \wedge (w)$. It is not hard to see that C is satisfied iff 13 clauses of $\mathcal{F}_C$ are satisfiable. ◀

The reduction: Set $\delta = 1$ and construct a graph G from $G_{\mathcal{F}}$ as follows. G consists of two types of gadgets; *variable* and *clause* gadgets. A variable gadget X_i has two vertices T_i and F_i representing the two possible assignments to X_i. A clause gadget of $(X_i \vee X_j)$ connects two pairs of vertices of two variable gadgets, such that a clause can be simplified with exactly two links if and only if it is satisfied by the corresponding assignment; see Figure 1.



(a) $(\bar{X}_i \vee X_j)$



(b) $(X_i \vee X_j)$

Figure 1 The clause gadgets. (a) is a clause gadget for $(\bar{X}_i \vee X_j)$. Note that any 2-link path from X_i to X_j simplifies the gadget, except for the path from T_i to F_j which corresponds to an assignment that does not satisfy the clause. In the case that the simplification uses this 2-link path, an additional edge should be added (the red subgraph). (b) is a clause gadget for $(X_i \vee X_j)$, and thus using the 2-link path from F_i to F_j requires and additional link. The construction for $(X_i \vee \bar{X}_j)$ and $(\bar{X}_i \vee \bar{X}_j)$ is symmetric.

Let $G_{\mathcal{F}} = (\mathcal{X}_1 \dot{\cup} \mathcal{X}_2, E)$, so $\mathcal{X}_1 \dot{\cup} \mathcal{X}_2$ is a partition of the variables and $G_{\mathcal{F}}$ is bipartite. Let ℓ_1 and ℓ_2 be two lines parallel to the x-axis, and vertical distance L. We place the variable gadgets belonging to $\mathcal{X}_1$ on ℓ_1 and those belonging to $\mathcal{X}_2$ on ℓ_2. Since $G_{\mathcal{F}}$ is bipartite, each clause gadget connects a variable from ℓ_1 to another variable from ℓ_2 (see Figure 2).

Now, we need to make sure that the edges of a clause gadget can be used (in the simplification subgraph) only to "cover" the edges of that clause. For this, we choose a sufficiently large L, and the variable gadgets are spaced apart at distance larger than $3 + \varepsilon$ along ℓ_1 and ℓ_2. See the full version of the paper for further details.

▶ **Theorem 2.2.** *The minimum-edge graph-graph simplification under the graph and traversal distances is NP-hard.*

3 Leaf-Restricted Tree-to-Tree Simplification from Input to Output

We now consider the case where 1-degree vertices of the input graph G are mapped to 1-degree vertices of the output simplification G'. Note that this restriction might be helpful

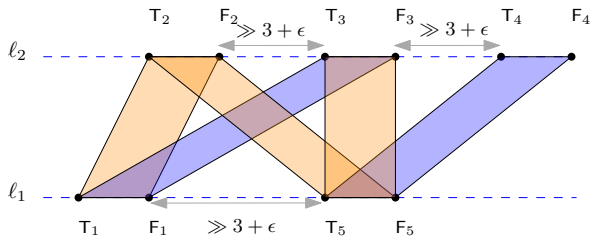


Figure 2 A schematic representation of the thickened embedded variable graph induced by $(X_1 \vee X_2) \wedge (X_1 \vee \bar{X}_3) \wedge (\bar{X}_2 \vee \bar{X}_5) \wedge (X_4 \vee \bar{X}_5) \wedge (X_3 \vee X_5)$. In this embedding, the type (a) and type (b) clause gadgets are highlighted in orange and blue strips respectively.

in capturing the structure of the graph (see e.g. [1]). We show that given a graph G embedded in $\mathbb{R}^2$ and a threshold δ, it is NP-hard to find a graph G' with minimum number of vertices such that $\overrightarrow{\delta}_G(G, G') \leq \delta$, and every 1-degree vertex of G is matched to a 1-degree vertex of G'. We call such a simplification G' a *leaf-restricted* simplification. The following theorem actually proves a stronger claim: finding a leaf-restricted simplification is NP-hard even for the case where the input is a tree T, and even if we restrict our simplification to be a tree. In addition, with a similar proof we can show that the problem under traversal distance is NP-hard. More details will be given in the full version of this paper.

▶ **Theorem 3.1.** *Given a tree T embedded in $\mathbb{R}^2$ and distance δ, finding a leaf-restricted simplification for T under graph distance with minimum number of vertices is NP-Hard.*

Proof Sketch. The proof is by a reduction from the unit disk cover problem (UDC): let P be a finite set of points in $\mathbb{R}^2$, the goal is to find a minimal set $\mathcal{D}$ of unit disks such that for any $p \in P$ there exists some $D \in \mathcal{D}$ that contains p. Given an instance P of UDC, let $\mathcal{R}(P)$ be the bounding box of P. Set $\delta = 1$, and construct a tree T as follows: $V(T) = P \cup \{x\}$ for some point x which lies far enough from $\mathcal{R}(P)$, and $E(T) = \{\langle px \rangle \mid p \in P\}$. We will show that there exists a solution $\mathcal{D}$ for P with k disks, if and only if there exists a leaf-to-leaf simplification T' for T under graph distance with $k + 1$ vertices.

$\Rightarrow$: Let $c_1, \ldots, c_k$ be the centers of disks in $\mathcal{D}$. Set $V(T') = \{x, c_1, \ldots, c_k\}$, and $E(T') = \{\langle c_i x \rangle \mid 1 \leq i \leq k\}$. Consider a mapping s that maps x to x and each vertex $p \in V(T)$ to some vertex $c_i \in V(T')$ such that p is contained in the unit disk centered at c_i. Now each edge $(x, p) \in E(T)$ is mapped to an edge $(x, c_i) \in E(T')$, and thus the distance is at most 1.

$\Leftarrow$: Let s be the mapping function from T to T'. Let $C = \{s(p) \mid p \in V(T) \setminus \{x\}\}$, then $C \subseteq V(T')$ because T' is a leaf-restricted simplification and thus s matches each leaf p of T to a leaf $s(p)$ of T'. Also, C contains at most k vertices because x is far enough from $\mathcal{R}(P)$, so there must be at least one vertex of T' which lies outside $\mathcal{R}(P)$ and such that none of the points of P is matched to it. Consider the unit disks centered around the points in C. Clearly, $\mathcal{D}$ covers P. ◀

4 Leaf-Restricted Tree-to-Tree Simplification from Output to Input

In this section, we consider the (strong or weak) graph distance from output to input and for a given set $l = \{l_1, \cdots, l_k\}$ of leaves with $k \geq 2$ in the input tree T, we require that the mapping of the graph distance bijects k distinct leaves of T' to the leaf set l in T. In this section T' selects its vertices from $V(T)$. Without any restriction on the leaves, the problem is trivial, as T' could consist of a single point.

Now, consider the complete graph G induced by $V(T)$. The intersection of G with a ball of radius δ centered at a vertex of $V(T)$ consists of several connected components. For a vertex $u \in T$, let C_u be the set of all edges of connected components induced by the intersection of G with the ball centered at u. Given a leaf set $l = \{l_1, \cdots, l_k\}$ in T, observe that the problem for $k = 2$ is simply the simple path from the respective leaf to the **root** which is a leaf in l. Relying on this observation we first compute the path from every leaf to the root in T. The union of these paths forms the *mapping subtree* M_T. A vertex where multiple paths meet in M_T is called an *ancestor*. We associate a pointer with each leaf to point to the lowest common ancestor obtained earlier and we repeat the same process for the new ancestors towards the root. The ancestors connected through pointers together result in a tree called *ancestor tree* A_T. An ancestor u is *parent* of v (v is child) if it is pointed to by v in A_T. The idea is to use DP to propagate the optimal simplified tree rooted at each edge of every connected component c' of every child to the connected component c of the parent. For this, we associate a cost function $\psi : [0, 1] \to \mathbb{N}$ with each edge e, where $\psi(e)$ is the number of vertices in a minimum-edge simplified subtree rooted at e. For a connected component c w.r.t. the parent u and c' w.r.t. child v, we have the following recursive formula:

$$\psi(e) = \min_{\mathcal{E}} \sum_{e' \in \mathcal{E}} \big(\psi(e') + \gamma(e', e)\big), \tag{1}$$

where $\mathcal{E} = \{\mathcal{E}_1, \mathcal{E}_2, \cdots, \mathcal{E}_{k'}\}$ with each edge $\mathcal{E}_i$ belonging to the edge set C_i, for all $1 \le i \le k'$ where k' is the number of children of u. Here, $\gamma(e', e)$ is the number of vertices on the vertex-restricted minimum-vertex curve simplification of the path $P[u, v] \in M_T$ computed in [9]. Also for every leaf $u \in l$ we set $\psi(e) = 1$. Figure 3 elaborates on the algorithm described above.

▶ **Lemma 4.1.** *Formula (1) returns the number of vertices in an optimal simplified tree T'.*

Proof. We use proof by induction. Suppose u is a leaf where $u \in l$. Obviously T' rooted at e is a single-vertex tree. Therefore $\psi(e) = 1$. When u is an intermediate vertex, suppose T^*_e is an optimal simplified tree rooted at e. Observe that T^*_e passes through a set of edges $\mathcal{E} = \{\mathcal{E}_1, \cdots, \mathcal{E}_{k'}\}$ with $\mathcal{E}_i \in E_i$. Therefore we have: $\psi(e) = \min_{T'_e} \mathsf{W}(T'_e) = \mathsf{W}(T^*_e) = \sum_{e' \in \mathcal{C}} (\mathsf{W}(T^*_{e'}) + \mathsf{W}(P(e', e)))$, where P is the minimum-vertex path between e and e'. Note that $\mathsf{W}(P(e', e)) = \gamma(e', e)$ by definition and $\mathsf{W}(T^*_{e'}) = \psi(e')$ by the inductive hypothesis. Thus we have: $\psi(e) = \mathsf{W}(T^*_e) = \sum_{e' \in \mathcal{E}} \big(\psi(e') + \gamma(e', e)\big)$. Realize that missing any of the connected components in $\mathcal{E}$ results in $\overrightarrow{\delta_G}(T', T) > \delta$. We therefore have: $\psi(e) = \mathsf{W}(T^*_e) = \min_{\mathcal{E}} \sum_{e' \in \mathcal{E}} \big(\psi(e') + \gamma(e', e)\big)$. ◀

▶ **Theorem 4.2.** *There exists an $O(kn^5)$ time algorithm for the leaf-to-leaf tree simplification from T' to T under the weak and strong graph distances, where T' selects its vertices in $\mathbb{R}^d$.*

Proof. First realize that $|C_u| = |E(G)| = O(n^2)$. Also computing M_T takes $O(kn)$ time. The only remaining part is that to compute $\psi(e)$ for all edges e of all connected components $c \in u$ and all ancestors $u \in M_T$. Since $|C_u| = O(n^2)$ and there are $O(k)$ vertices like u in M_T, thus there are $O(kn^2)$ starting points like e to compute $\gamma(e, e')$ for. Computing $\gamma(e, e')$ takes $O(n^3)$ for all e' following [9]. Overall, the algorithm takes $O(kn^5)$. ◀

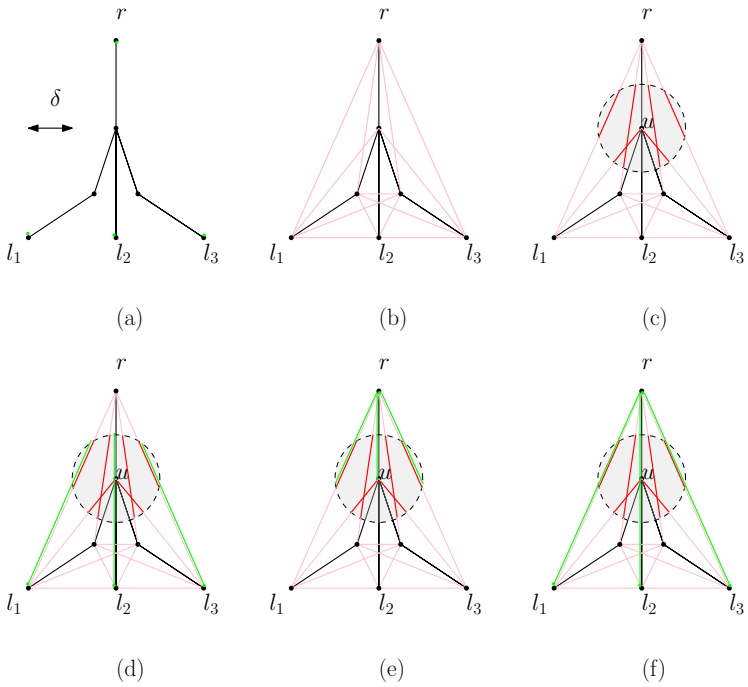


Figure 3 An illustration of the algorithm. (a) The input tree and the threshold δ. Here, $k = 3$. (b) The shortcut graph (complete graph) G highlighted in pink. (c) Computing common ancestor of the leaves, identifying the connected components of C_u highlighted in red. (d) Min-link path simplification of $P[l_i, u]$ highlighted in green, starting at l_i and ending at all combinations of all edges in the connected components of C_u, for all $i = 1, 2, 3$. (e) The continuation of the min-link path simplification of $P[u, r]$ starting at edges in C_u and ending at r. (f) The resulting min-vertex simplifiied tree in green.

References

1 H.A. Akitaya, M. Buchin, B. Kilgus, S. Sijben, and C. Wenk. Distance measures for embedded graphs. In *Proc. 30th Int. Symp. on Algorithms and Computation*, pages 55:1–55:15, 2019. doi:10.4230/LIPIcs.ISAAC.2019.55.

2 H. Alt, A. Efrat, G. Rote, and C. Wenk. Matching planar maps. *Journal of Algorithms*, 49(2):262—283, 2003.

3 E. Bindewald and B.A. Shapiro. RNA secondary structure prediction from sequence alignments using a network of k-nearest neighbor classifiers. *RNA*, 12(3):342–352, 2006.

4 P. Cignoni, C. Montani, and R. Scopigno. A comparison of mesh simplification algorithms. *Computers & Graphics*, 22(1):37–54, 1998.

5 R. Estkowski and J.S.B. Mitchell. Simplifying a polygonal subdivision while keeping it simple. In *Proc. 17th Ann. ACM Symp. Computational Geometry*, pages 40–49, 2001.

6 S. Funke, T. Mendel, A. Miller, S. Storandt, and M. Wiebe. Map simplification with topology constraints: Exactly and in practice. In *Proc. 19th Workshop on Algorithm Engineering and Experiments*, pages 185–196, 2017.

7 J. Gudmundsson, P. Laube, and T. Wolle. Movement patterns in spatio-temporal data. In S. Shekhar and H. Xiong, editors, *Encyclopedia of GIS*. Springer-Verlag, 2007.

8 L. Guibas, J. Hershberger, J. Mitchell, and J. Snoeyink. Approximating polygons and

subdivisions with minimum-link paths. *International Journal of Computational Geometry & Applications*, 3(4):383–415, 1993.

9 M. van de Kerkhof, I. Kostitsyna, M. Löffler, M. Mirzanezhad, and C. Wenk. Global curve simplification. In *27th Ann. European Symposium on Algorithms*, volume 144, pages 1–14, 2019.

10 Y. Lee. Handwritten digit recognition using k nearest-neighbor, radial-basis function, and backpropagation neural networks. *Neural Computation*, 3(3):440–449, 1991.

11 T. Mendel. Area-preserving subdivision simplification with topology constraints: Exactly and in practice. In *Proc. 20th Workshop on Algorithm Engineering and Experiments*, pages 117–128, 2018.

12 W. Meulemans. *Similarity Measures and Algorithms for Cartographic Schematization*. PhD thesis, Eindhoven University of Technology, 2014.

On the Geometric Red-Blue Hitting Set Problem

Raghunath Reddy Madireddy[1] and Supantha Pandit[2]

1 **Birla Institute of Technology and Science, Pilani- Hyderabad campus, India**
raghunath@hyderabad.bits-pilani.ac.in
2 **Dhirubhai Ambani Institute of Information and Communication Technology, Gandhinagar, Gujarat, India**
pantha.pandit@gmail.com

Abstract

We study variations of the Red-Blue Hitting Set (*RBHS*) problem. We first introduce a variation of this problem in a set system, the Special-Red-Blue Hitting Set (*SPECIAL-RBHS*) problem. We prove that, the *SPECIAL-RBHS* problem is APX-hard. Next, using this result, we show that the *RBHS* problem is APX-hard for the following classes of objects in the plane: (i) rectangles containing the origin of the plane, (ii) downward shadows of segments, (iii) axis-parallel strips, and (iv) axis-parallel rectangles where every two rectangles intersect exactly either zero or four times. Further, we prove that the *RBHS* problem is NP-hard for: (i) rectangles anchored on two parallel lines and (ii) rectangles intersect a horizontal line. We also show that the *RBHS* problem can be solved in polynomial time when the objects are axis-parallel infinite lines. On the contrary, we prove that the *RBHS* problem is NP-hard when the objects are vertical lines and horizontal unit segments. The latter result also ensures that the *RBHS* problem with axis-parallel unit segments is NP-hard.

1 Introduction

We study the following variation of the Hitting Set problem in a geometric setting.

> **Red-Blue Hitting Set (*RBHS*) Problem**. We are given a set of points P in the plane and two sets of objects R (red objects) and B (blue objects). The goal is to pick a subset $P' \subseteq P$ of points that hits all blue objects in B while hitting the minimum number of red objects in R.

Researchers have studied variations of the Set Cover problem due to its numerous applications and one of such variations is the Red-Blue Set Cover problem (*RBSC*). The problem was first introduced by Carr et al. [2] in a set system and proved that, unless P=NP, the problem cannot be approximated within $2^{\log^{1-\delta} n}$ factor in polynomial-time, where $\delta = \frac{1}{\log^c \log n}$, for any constant $c \leq \frac{1}{2}$ and n is the number of sets in the set system. Chan and Hu [4] first considered *RBSC* problem in geometric setting and proved that the problem is NP-hard for axis-parallel unit squares and provided a PTAS for the same. Further, it is known that the *RBSC* problem is APX-hard where the objects are axis-parallel rectangles [10]. Dom et al. [7] studied the *RBHS* problem (in a set system) with consecutive ones property. Based on whether the red set or blue set, or both satisfy the consecutive ones property, they gave either NP-hardness proof or polynomial-time algorithm. In [5], Chang et al. considered the same problem and gave an improved polynomial-time algorithm for the *RBHS* problem when both red and blue sets satisfy the consecutive one property.

1.1 Our contributions

⇨ We define a variation of the *RBHS* problem on set systems, the *SPECIAL-RBHS* problem, and show that it is APX-hard. Using this result, we show that the *RBHS* problem is

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

APX-hard for several classes of objects (see the list in Theorem 2.2) (Section 2).

- ⇨ We prove that the *RBHS* problem is NP-hard when the objects are (i) rectangles anchored on two parallel lines and (ii) rectangles intersecting a horizontal line (Section 3).
- ⇨ We show that the *RBHS* problem with axis-parallel lines is polynomial-time solvable. We also show that the problem is NP-hard for vertical lines and horizontal unit segments. The reduction implies that the problem is NP-hard for axis-parallel unit segments (Section 4).

1.2 Preliminaries

Monotone 3-SAT (*M3SAT*) problem. In a CNF formula, a clause is said to be *negative* if all its literals are negatively present in that clause. Otherwise, if all it's literal are positively present, then the clause is called a *positive* clause. In *Monotone 3-SAT (M3SAT)* problem, a 3-CNF formula ϕ is given, where each clause is either positive or negative and contains exactly three literals, the objective is to decide whether ϕ is satisfiable.

It is known that the *M3SAT* problem is NP-complete [8]. Consider a planar embedding of the *M3SAT* problem, called the *Planar Monotone Rectilinear 3-SAT (PMR3SAT)* problem that is known to be NP-hard [6].

Planar Monotone Rectilinear 3-SAT (*PMR3SAT*) problem [6]. In this problem, for each variable or clause, a segment is considered. The variable segments are on a horizontal line L ordered from left to right. The positive (resp. negative) clause segments are placed below (resp. above) the line L and they are in different levels in the y-direction. Each clause connects to the three literals it contains by vertical connections. Finally, the embedding is drawn in such a way that it becomes planar. An instance of the problem is given in Figure 1.

2 APX-hardness Results for the *RBHS* Problem

We show that the *RBHS* problem is APX-hard for several classes of objects. First, we introduce the *SPECIAL-RBHS* problem and show that it is APX-hard. Next, we give an encoding for each class from an instance of the *SPECIAL-RBHS* problem.

Special-Red-Blue Hitting Set (*SPECIAL-RBHS*) Problem:
Let (U, X) be a range space, and $U = S \cup T$, where $S = \{s_i^1, s_i^2, s_i^3 \mid i = 1, 2, \ldots, n\}$ and $T = \{t_p^1, t_p^2, t_p^3, t_p^4 \mid p = 1, 2, \ldots, m\}$ are the sets of elements. Further, $X = R \cup B$, where R and B are the collections of, red and blue, subsets of U, respectively, such that

1. Each element in U belongs to exactly one set in R and exactly one set in B.
2. For each $i = 1, 2, \ldots, n$, the set R contains a subset $\{s_i^1, s_i^2, s_i^3\} \subseteq S$.
3. For each $p = 1, 2, \ldots, m$, there exist two integers i and j, where $1 \leq i < j \leq n$, such that X contains five subsets $\{s_i^k, t_p^1\}$, $\{t_p^1, t_p^2\}$, $\{t_p^2, t_p^3\}$, $\{t_p^3, t_p^4\}$, and $\{t_p^4, s_j^{k'}\}$ of U for some $k, k' \in \{1, 2, 3\}$. Further, the sets $\{s_i^k, t_p^1\}$, $\{t_p^2, t_p^3\}$, and $\{t_p^4, s_j^{k'}\}$ are in B and the sets $\{t_p^1, t_p^2\}$ and $\{t_p^3, t_p^4\}$ are in R.

The goal is to find a subset $U^* \subseteq U$ of elements that hits all the sets in B while hitting the minimum number of sets in R.

▶ **Theorem 2.1.** *The SPECIAL-RBHS problem is* APX*-hard.*

Proof. We give an L-reduction [9] from an APX-hard problem, the *vertex cover problem on cubic graphs* [1] to the *SPECIAL-RBHS* problem. Let $G = (V, E)$ be a cubic graph

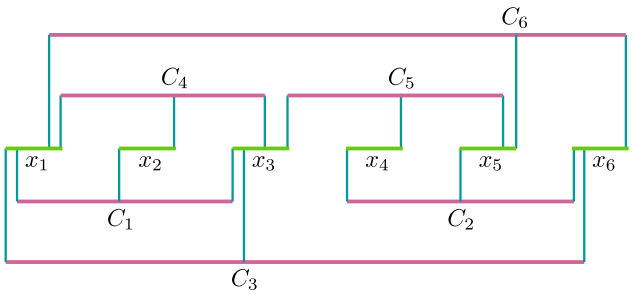


Figure 1 An instance of the Rectilinear Planar Monotone 3-SAT problem.

where $V = \{v_1, v_2, \ldots, v_n\}$ and $E = \{e_1, e_2, \ldots, e_m\}$. We generate an instance (U, X) of the *SPECIAL-RBHS* problem as follows:

1. For each vertex $v_i \in V$, take three elements s_i^1, s_i^2, s_i^3 in S. Thus, $S = \{s_i^1, s_i^2, s_i^3 \mid i = 1, 2, \ldots, n\}$. Further, for each vertex $v_i \in V$, place a set $\{s_i^1, s_i^2, s_i^3\}$ in R.
2. For each edge $e_p \in E$, consider four elements t_p^1, t_p^2, t_p^3, and t_p^4 in T. Thus, $T = \{t_p^1, t_p^2, t_p^3, t_p^4 \mid p = 1, 2, \ldots, m\}$.
3. For each edge $e_p = (v_i, v_j) \in E$ where $1 \leq i < j \leq n$ do the following:
   a. Let k be a positive integer, $1 \leq k \leq 3$, (resp. k', $1 \leq k' \leq 3$) such that e_p be the k-th (resp. k') edge incident on v_i (resp. v_j) in the order $e_1, e_2, \ldots, e_m$.
   b. Consider five subsets $\{s_i^k, t_p^1\}$, $\{t_p^1, t_p^2\}$, $\{t_p^2, t_p^3\}$, $\{t_p^3, t_p^4\}$, $\{t_p^4, s_j^{k'}\}$ of $S \cup T$. Place the sets $\{s_i^k, t_p^1\}$, $\{t_p^2, t_p^3\}$, $\{t_p^4, s_j^{k'}\}$ in B and place sets $\{t_p^1, t_p^2\}$ and $\{t_p^3, t_p^4\}$ in R.
4. Finally, we have $U = S \cup T$ and $X = R \cup B$.

Let $V^* \subseteq V$ be an optimal vertex cover of G and $U^* \subseteq U$ be an optimal hitting set for the instance of the *SPECIAL-RBHS* problem i.e., the elements in U^* hit all sets in B but hit the minimum number of sets in R. Let R^* be the set of subsets of R that are hit by elements in U^*. In the following, we first show that $|R^*| = |V^*| + |E|$. Initially, let $U^* = \emptyset$.

Let $e_p = (v_i, v_j)$ be an edge in the graph G where $1 \leq i < j \leq n$. To cover the edge e_p, at least one of v_i and v_j is in V^*. There are three possible cases:

(i) Both $v_i, v_j \in V^*$: Pick both s_i^k and $s_j^{k'}$ into U^* and exactly one of t_p^2 and t_p^3 into U^*.
(ii) $v_i \in V^*$ and $v_j \notin V^*$: Pick s_i^k into U^*. Further, pick both t_p^3 and t_p^4 into U^*.
(iii) $v_i \notin V^*$ but $v_j \in V^*$: Pick $s_j^{k'}$ into U^*. Further, pick both t_p^1 and t_p^2 into U^*.

In each case above, the elements picked from T into U^* hit exactly one set in R and further these sets do not hit by any element picked in U^* from set S. By following the above procedure for all the edges in G, we can obtain an optimal solution $U^* \subseteq U$ for the instance of *SPECIAL-RBHS* problem with $|R^*| = |V^*| + |E|$. Since G is a cubic graph, $|V^*| \geq \frac{1}{3}|E|$. Thus, $|R^*| \leq 4|V^*|$. Hence, in the definition of L-reduction, $\alpha = 4$.

Let $U_1 \subseteq U$ be a solution to the *SPECIAL-RBHS* problem. Let $R_1 \subseteq R$ be the set of subsets that are hit by elements in U_1. Note that one can obtain a vertex cover $V_1 \subseteq V$ of size at most $|R_1| - |E|$. Therefore, $||V_1| - |V^*|| \leq ||R_1| - |E| - |V^*|| \leq ||R_1| - |R^*||$, implies $\beta = 1$ in definition of L-reduction. Therefore, *SPECIAL-RBHS* is APX-hard. ◀

▶ **Theorem 2.2.** *The RBHS problem is* APX*-hard for* ***(O1)*** *Axis-parallel rectangles where each pair of rectangles intersect exactly zero or four times,* ***(O2)*** *Axis-parallel strips,* ***(O3)*** *Rectangles containing the origin of the plane and* ***(O4)*** *Downward shadows of segments.*

Proof. **O1:** We place the points $s_1^1, s_1^2, s_1^3, s_2^1, s_2^2, s_2^3, \ldots, s_n^1, s_n^2, s_n^3$ on a horizontal line in the same order from left to right such that for each $i = 1, 2, \ldots, n$, s_i^1, s_i^2, and s_i^3 are close to each other (Figure 2). We place a rectangle that covers only s_i^1, s_i^2, and s_i^3 (Figure 3a). There is a sufficient gap between the points corresponding to different i's (Figure 2) and this gap is used to place the points t_p^2 and t_p^3, for $p = 1, 2, \ldots, m$. Further, for each $p = 1, 2, \ldots, m$, there is a dedicated region in which we place the points t_p^1, t_p^2, t_p^3, and t_p^4 (Figure 2). The placement of t_p^1, t_p^2, t_p^3, and t_p^4 and the respective rectangles is shown in Figure 3a.
O2: The encoding is similar to class O1 (Figure 3b) except the placement of $s_1^1, s_1^2, s_1^3, \ldots, s_n^1, s_n^2, s_n^3$. For each $i = 1, 2, \ldots, n$, the tuples of points s_i^1, s_i^2, and s_i^3 are placed in an increasing stair-case fashion from bottom to top. The placement of other points is the same as in the case of O1.
The encoding for **O3** and **O4** is similar to the embedding in [3] (see Figure 3c and 3d). ◀

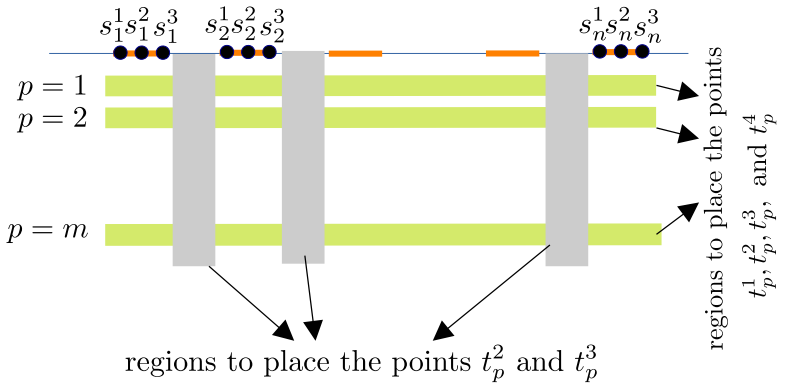


Figure 2 Outline of placement of points.

3 The *RBHS* Problem: Rectangles anchored on two parallel lines

We prove that the *RBHS* problem with rectangles anchored on two parallel lines (*RBHS-RATL* problem) is NP-hard by giving a reduction from *PMR3SAT* problem. Let ϕ be an instance of the *PMR3SAT* problem with n variables $x_1, x_2, \ldots, x_n$ and m clauses $C_1, C_2, \ldots, C_m$. Below we construct an instance H_ϕ of the *RBHS-RATL* problem from ϕ as follows.
Variable gadget: The gadget for the variable x_i is shown in Figure 4. Let L_1 and L_2 be two horizontal lines. There are $4m + 2$ points $\{p_1^i, p_2^i, \ldots, p_{4m+2}^i\}$ that form a circular structure. Also, there are $4m+2$ blue rectangles $\{b_1^i, b_2^i, \ldots, b_{4m+2}^i\}$ and $4m+2$ red rectangles $\{r_1^i, r_2^i, \ldots, r_{4m+2}^i\}$. The $2m+1$ blue rectangles $\{b_1^i, b_2^i, \ldots, b_{2m+1}^i\}$ and $2m+1$ red rectangles $\{r_1^i, r_2^i, \ldots, r_{2m+1}^i\}$ are anchored on L_1 and the remaining blue and red rectangles are anchored on L_2. The blue rectangle b_j^i covers two points p_{j-1}^i and p_j^i, for $1 \leq j \leq 4m + 2$ (assuming p_0^i as p_{4m+2}^i) and the red rectangle r_j^i covers the point p_j^i, for $1 \leq j \leq 4m + 2$. It is clear that there are two sets; $P_1^i = \{p_1^i, p_2^i, \ldots, p_{4m+1}^i\}$ and $P_2^i = \{p_2^i, p_4^i, \ldots, p_{4m+2}^i\}$; such that each of them hits all the blue rectangles and minimum number $(2m + 1)$ of red rectangles. The set P_1^i represents that the variable x_i is false and the set P_2^i represents that x_i is true.
Clause gadget: For each clause gadget, we take a blue rectangle. If the clause is positive (resp. negative), then the top (resp. bottom) boundary of the rectangle coincides with the clause segment, and the bottom (resp. top) boundary coincides with L_1 (resp. L_2). See Figure 5 for a schematic diagram of clause gadgets and its placement with respect to the variable gadgets for the instance of the *PMR3SAT* problem in Figure 1.

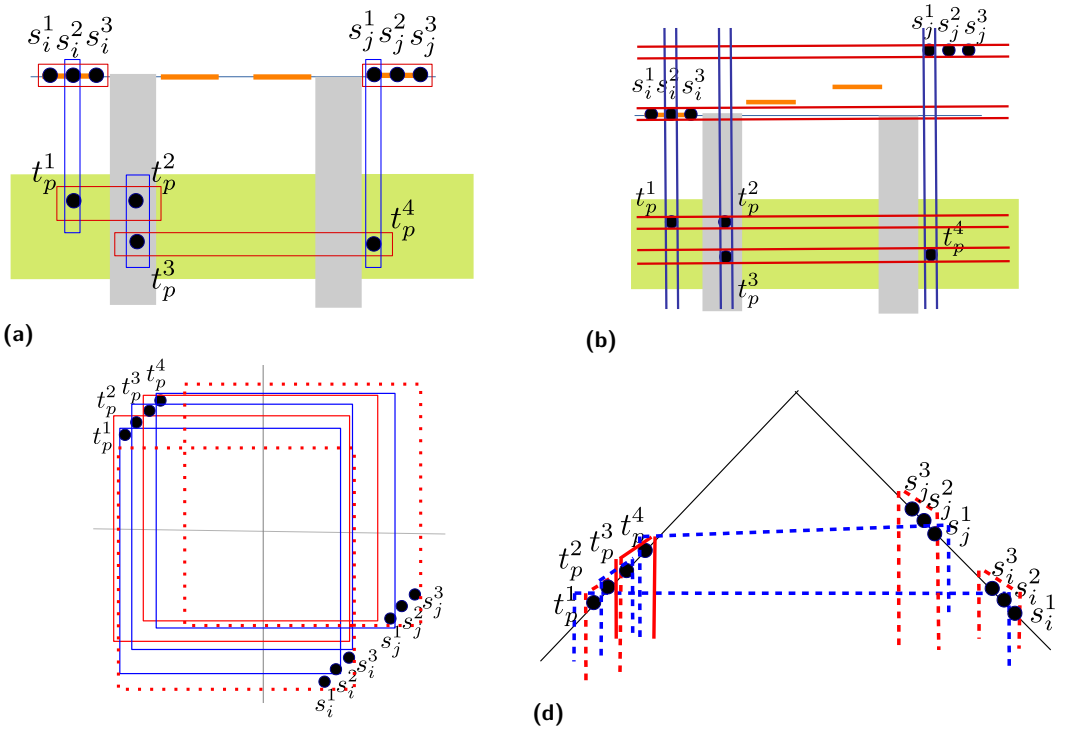


Figure 3 The encoding of sets $\{s_i^k, t_p^1\}$, $\{t_p^1, t_p^2\}$, $\{t_p^2, t_p^3\}$, $\{t_p^3, t_p^4\}$, and $\{t_p^4, s_j^{k'}\}$ for $k = 2$ and $k' = 1$ for classes, (a) Class O1, (b) Class O2, (c) Class O3, and (d) Class O4.

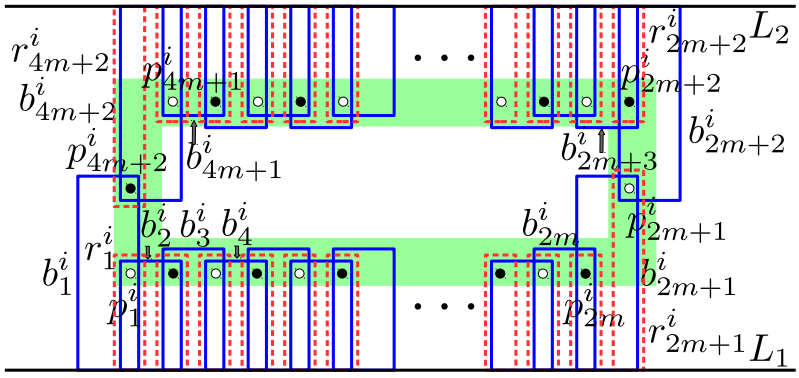


Figure 4 Structure of a variable gadget

Placement of clause gadgets with respect to variable gadgets: There are two lines L_1 and L_2. Each variable segment is replaced with a variable gadget and each clause segment is replaced with a rectangle. Now we describe the placement of the clause gadgets. For variable x_i, order the connections of positive clauses left to right that connect x_i. Let C_ℓ be a positive clause that connects to x_i through the t-th connection according to this order, then we say that C_ℓ is the t-th clause for x_i. A similar description can be given for negative clauses by looking at the *PMR3SAT* embedding (Figure 1) upside down.

Let C_ℓ be a positive clause that contains x_i, x_j, and x_k. Further, assume that C_ℓ is the t_1-th, t_2-th, and t_3-th clause for variables x_i, x_j, and x_k, respectively. Then, we shift

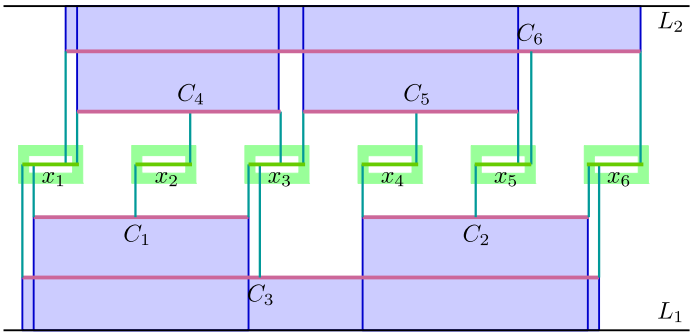


Figure 5 A schematic structure of the variable and clause gadgets and their interconnection.

the three point $p^i_{2t_1}$, $p^j_{2t_2}$, and $p^k_{2t_3}$ vertically downwards from the variables x_i, x_j, and x_k, respectively, along the top boundary of b^ℓ (see Figure 6).

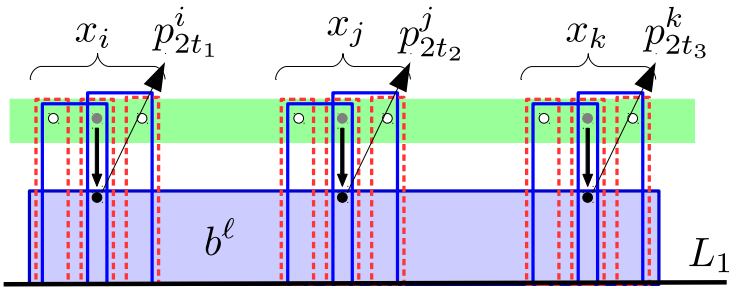


Figure 6 A clause gadget and its interaction with variable gadgets.

It is clear that the construction can be done in polynomial-time (in n and m).

▶ **Theorem 3.1.** *The RBHS-RATL problem is* NP*-hard.*

A modification to above construction ensures that the *RBHS* problem with rectangles stabbing a horizontal line (*RBHS-RSHL* problem) is NP-hard.

4 The *RBHS* problem with Lines and Segments

We first note that the *RBHS* problem with infinite axis-parallel lines can be solved in polynomial time by reducing the problem to the weighted edge cover problem in bipartite graphs. In the below, we prove that the *RBHS* problem with vertical lines and horizontal segments (*RBHS-VLHS* problem) is NP-hard. We give a reduction from the *M3SAT* problem. We generate an instance H_ϕ of the *RBHS-VLHS* problem from an instance ϕ of the *M3SAT* problem as follows.

Variable gadget: For the variable x_i, the gadget (Figure 7) consists of $4m+4$ blue segments $\{b^i_1, b^i_2, \ldots, b^i_{4m+4}\}$, $4m+4$ red segments $\{r^i_1, r^i_2, \ldots, r^i_{4m+4}\}$, and $4m+4$ points $\{p^i_1, p^i_2, \ldots, p^i_{4m+4}\}$. The $2m+1$ blue segments $\{b^i_2, b^i_3, \ldots, b^i_{2m+2}\}$, $2m+2$ red segments $\{r^i_1, r^i_2, \ldots, r^i_{2m+2}\}$, and $2m+2$ points $\{p^i_1, p^i_2, \ldots, p^i_{2m+2}\}$ are on a horizontal line. The $2m+1$ blue segments $\{b^i_{2m+4}, b^i_{2m+5}, \ldots, b^i_{4m+4}\}$ and the $2m+2$ red segments $\{r^i_{2m+3}, r^i_{2m+4}, \ldots, r^i_{4m+4}\}$, and $2m+2$ points $\{p^i_{2m+3}, p^i_{2m+4}, \ldots, p^i_{4m+4}\}$ are another horizontal line. There are two blue vertical lines b^i_1 and b^i_{2m+3}. We place the blue and red segments, and points in such a way that the point p^i_j hits exactly two blue segments b^i_j and b^i_{j+1} and exactly one red

segment r^i_j, for $1 \leq j \leq 4m+4$ (assuming $b^i_{4m+5} = b^i_1$) (Figure 7). Observe that, there are exactly two sets of points; $P^i_1 = \{p^i_1, p^i_3, \ldots, p^i_{4m+3}\}$ and $P^i_2 = \{p^i_2, p^i_4, \ldots, p^i_{4m+4}\}$ such that each of the set of points hits all the blue segments and half (i.e., $2m+2$) of the red segments. We assume that, the set P^i_1 interprets x_i to be false and P^i_2 interprets x_i to be true.

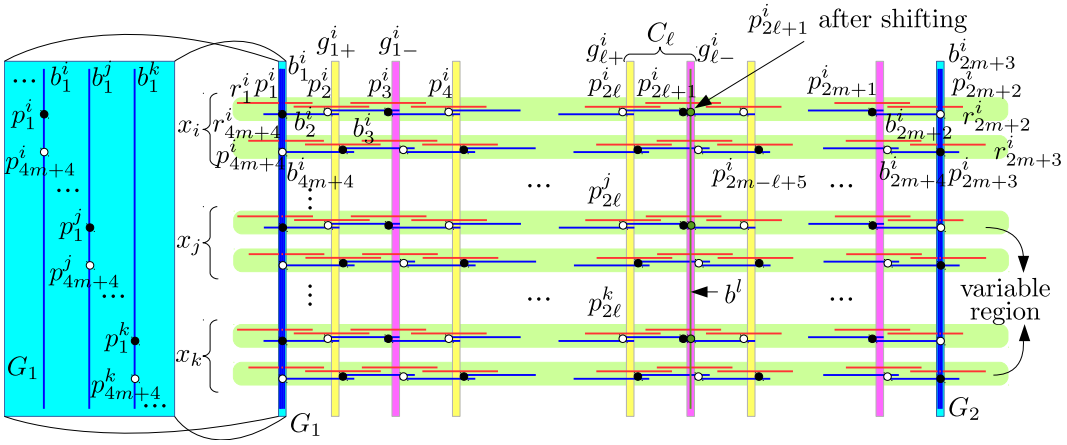


Figure 7 Schematic construction of an instance of the *RBHS-VLHS* problem from an instance of the *PMR3SAT* problem. Here, the strip G_1 is magnified at the left side of the figure. A similar structure as G_1 is there inside the strip G_2.

Overall structure: The overall structure of the construction is shown in Figure 7. The variable gadgets are placed vertically one after another in an identical way (i.e., all the j-th point; except 1-st, $(2m+2)$-th, $(2m+3)$-th, and $(4m+4)$-th; from n variables are on a vertical line, and left end points of j-th segment, except 1-st and $(m+3)$-th), are also on a vertical line. There are two regions G_1 and G_2 at the extreme left and extreme right of the construction. In the region G_1, m blue vertical lines $\{b^1_1, b^2_1, \ldots, b^m_1\}$ are arranged in a order from left to right. Similarly, in G_2, m blue vertical lines $\{b^1_{2m+3}, b^2_{2m+3}, \ldots, b^m_{2m+3}\}$ are arranged in a order from left to right. To the right of the point $p^i_{2\ell}$ (resp. $p^i_{2\ell+1}$) there is a dedicated region g^ℓ_+ (resp. g^ℓ_-) where the gadget of C_ℓ is placed, $1 \leq \ell \leq m$.

Clause gadgets and its placement: Let C_ℓ be a negative clause that contains variables x_i, x_j, and x_k. For C_ℓ, the gadget consists of a single vertical line b^ℓ. The line b^ℓ is placed inside the region g^ℓ_-. The 3 points $p^i_{2\ell+1}$, $p^j_{2\ell+1}$, and $p^k_{2\ell+1}$ are shifted horizontally and placed inside g^ℓ_- (see Figure 7). If C_ℓ is positive the construction is similar use the region g^ℓ_+.

This completes the construction and the construction can be made in polynomial (in n and m) time. We conclude the following theorem.

▶ **Theorem 4.1.** *The RBHS-VLHS problem is* NP*-hard.*

Axis-parallel unit segments: In the above construction the horizontal segments are unit length. If we vertically compress the construction then vertical lines becomes unit segment. Therefore, we conclude that the *RBHS* problem is NP-hard for axis-parallel unit segments.

References

1 Paola Alimonti and Viggo Kann. Some APX-completeness results for cubic graphs. *Theoretical Computer Science*, 237(1):123 – 134, 2000.

2 Robert D. Carr, Srinivas Doddi, Goran Konjevod, and Madhav Marathe. On the Red-blue Set Cover Problem. In *SODA*, pages 345–353, 2000.

3 Timothy M. Chan and Elyot Grant. Exact Algorithms and APX-hardness Results for Geometric Packing and Covering Problems. *Computational Geometry*, 47(2):112–124, 2014.

4 Timothy M. Chan and Nan Hu. Geometric red-blue set cover for unit squares and related problems. *Comput. Geom.*, 48(5):380–385, 2015.

5 Maw-Shang Chang, Hsiao-Han Chung, and Chuang-Chieh Lin. An improved algorithm for the red-blue hitting set problem with the consecutive ones property. *Inf. Process. Lett.*, 110(20):845–848, 2010.

6 Mark de Berg and Amirali Khosravi. Optimal binary space partitions for segments in the plane. *Int. J. Comput. Geom. Appl.*, 22(3):187–206, 2012.

7 Michael Dom, Jiong Guo, Rolf Niedermeier, and Sebastian Wernicke. Red-blue covering problems and the consecutive ones property. *J. Discrete Algorithms*, 6(3):393–407, 2008.

8 Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

9 Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, Approximation, and Complexity Classes. *Journal of Computer and System Sciences*, 43(3):425 – 440, 1991.

10 Sima Hajiaghaei Shanjani. Hardness of approximation for red-blue covering. In *CCCG*, 2020.

Intersecting Disks using Two Congruent Disks*

Byeonguk Kang[1], Jongmin Choi[1], and Hee-Kap Ahn[2]

1 **Department of Computer Science and Engineering, Pohang University of Science and Technology, Pohang, Korea.**
{kbu417,icothos}@postech.ac.kr

2 **Department of Computer Science and Engineering, Graduate School of Artificial Intelligence, Pohang University of Science and Technology, Pohang, Korea.**
heekap@postech.ac.kr

Abstract

We consider the Euclidean 2-center problem for a set of n disks in the plane: find two smallest congruent disks such that every disk in the set intersects at least one of the two congruent disks. We present a deterministic algorithm for the problem that returns an optimal pair of congruent disks in $O(n^2 \log^3 n / \log\log n)$ time. We also present a randomized algorithm with $O(n^2 \log^2 n / \log\log n)$ expected time. These results improve the previously best deterministic and randomized algorithms, making a step closer to the optimal algorithms for the problem.

1 Introduction

We consider a generalization of the 2-center problem [1, 4, 7, 8] in which given a set $\mathcal{D}$ of n disks of nonnegative radii in the plane, find two smallest congruent disks C_1 and C_2 satisfying $D \cap (C_1 \cup C_2) \neq \emptyset$ for every $D \in \mathcal{D}$. We call this problem the *2-center problem on disks.*

Ahn et al. [2] gave a deterministic algorithm for the problem with $O(n^2 \log^4 n \log\log n)$ time and a randomized algorithm with $O(n^2 \log^3 n)$ expected time. They showed that their algorithms also work for the *restricted 2-cover problem* on disks (every disk is contained in one of two smallest congruent disks) and the *2-piercing problem on disks* (every disk is pierced by one of two optimal points) in the plane. See Figure 1 for an illustration.

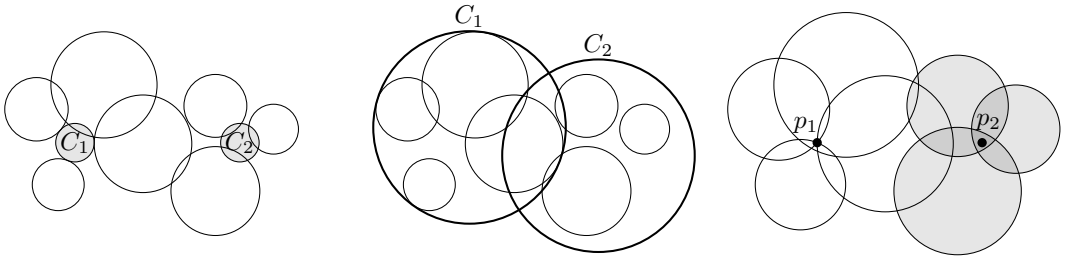


Figure 1 (a) The disk 2-center problem on disks: every input disk intersects $C_1 \cup C_2$. (b) The restricted disk 2-cover problem on disks: every disk is fully contained in C_1 or C_2. (c) The 2-piercing problem on disks: every disk intersects p_1 or p_2.

* This research was partly supported by the Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No. 2017-0-00905, Software Star Lab (Optimal Data Structure and Algorithmic Applications in Dynamic Geometric Environment)) and (No. 2019-0-01906, Artificial Intelligence Graduate School Program(POSTECH)).

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Our results. We present a deterministic algorithm with $O(n^2 \log^3 n/\log\log n)$ time and a randomized algorithm with $O(n^2 \log^2 n/\log\log n)$ expected time for the 2-center problem and the restricted 2-cover problem on n disks in the plane. This improves the previously best known algorithm by more than $O(\log n)$ factor. Our deterministic algorithm also works on the 2-piercing problem with $O(n^2 \log^2 n/\log\log n)$ time.

For a disk D, the disk inflated by real value $r \geq 0$ from D, denoted by $D(r)$, is centered at the center of D and its radius is the radius of D plus r. We use a dual arrangement $\mathcal{A}$ of the disk centers and construct a dual directed tree $T_{\mathcal{E}}$ of $\mathcal{A}$. Our sequential decision algorithm traverses the tree in directions of inserting inflated disks one by one and finds the centers. Our sequential decision algorithm works as follows.

1. Construct a point-line dual arrangement $\mathcal{A}$ of the disk centers such that each face of $\mathcal{A}$ represents the inflated disks whose centers lie in one side of a line in primal space.
2. Construct a directed tree $T_{\mathcal{E}}$ such that there is a one-to-one correspondence between the tree nodes and the faces of $\mathcal{A}$, and each edge is directed from a node to a neighboring node of lower level in $\mathcal{A}$.
3. Construct a collection $\mathcal{T}_t$ of t-ary search trees that, given a face f in $\mathcal{A}$, returns $O(\log n/\log\log n)$ regions whose common intersection is the intersection of the inflated disks represented by f.
4. Check for each face f while traversing $T_{\mathcal{E}}$ if the inflated disks represented by f have a nonempty intersection and the remaining inflated disks also have a nonempty intersection, using $\mathcal{T}_t$ and an insertion-only convex programming.

Our deterministic algorithm uses Cole's parametric search [5] with an $O(n^2 \log^2 n/\log\log n)$-time sequential decision algorithm and an $O(\log n)$-time parallel decision algorithm using $O(n^2 \log^2 n/\log^2 \log n)$ processors, after $O(n^2 \log^3 n/\log\log n)$-time preprocessing. The improvement of the sequential decision algorithm comes from the insertion-only convex programming and the data structure $\mathcal{T}_t$. The parallel decision algorithm constructs $\mathcal{T}_t$ in the preprocessing phase and the convex programming runs in parallel. Putting them together using Cole's parametric search, we get an $O(n^2 \log^3 n/\log\log n)$-time algorithm. A randomized algorithm with $O(n^2 \log^2 n/\log\log n)$ expected time can be obtained by combining our sequential decision algorithm and Chan's randomized optimization technique [3].

2 Preliminaries

▶ **Observation 2.1** (Observation 1 in [2]). *Let (C_1, C_2) be a pair of optimal covering disks. Let ℓ be the bisector of the segment connecting the centers of C_1 and C_2. Then, $C_i \cap D \neq \emptyset$ for every $D \in \mathcal{D}$ whose center lies on the same side of ℓ as the center of C_i, for $i = \{1, 2\}$.*

For a line ℓ in the plane, let B_ℓ be a bipartition of $\mathcal{D}$ to $\mathcal{D}_\ell$ and $\mathcal{D}_\ell^c = \mathcal{D} \setminus \mathcal{D}_\ell$, where $\mathcal{D}_\ell$ is the set consisting of disks in $\mathcal{D}$ with centers lying strictly below ℓ. Based on Observation 2.1, B_ℓ defines a subproblem consisting of two 1-center problems such that the smallest radius for the subproblem is the larger one of the two radii from the 1-center problems.

Given a real value $r \geq 0$, the decision 2-center problem on $\mathcal{D}$ is to determine whether $r \geq r^*$, where r^* is the radius of the two smallest congruent disks of the 2-center problem on $\mathcal{D}$. Let $\mathcal{D}(r)$ be the set of the inflated disks $D(r)$ of disks $D \in \mathcal{D}$. Then the decision 2-center problem on $\mathcal{D}$ with radius r reduces to the 2-piercing problem on $\mathcal{D}(r)$.

Given compact convex subsets in the plane, each representing a constraint, and an objective function, a point that satisfies the constraints and minimizes the objective function value can be found using convex programming. There are two types of primitive operations:

finding the leftmost feasible point of two constraints, and determining whether a given point is contained in a constraint. Convex programming can be used to determine whether the intersection of input convex sets is empty or not.

▶ **Lemma 2.2.** *A problem with k convex constraints can be solved in $O(T_c + \log k)$ time using convex programming, with $O(k^2)$ processors, where T_c denotes the time per primitive operation.*

▶ **Lemma 2.3.** *Given a convex program with k constraints and the leftmost point v^* of the intersection of the constraints, the operation of adding a new constraint to the convex programming can be handled such that the leftmost point in the intersection of the $k+1$ constraints can be found in $O(kT_c)$ time, where T_c denotes the time per primitive operation.*

In the following, we consider the 2-piercing problem on inflated disks in $\mathcal{D}(r)$.

3 The 2-piercing Problem on Disks

We present an algorithm for the 2-piercing problem on a set $\mathcal{E} = \{E_1, E_2, \ldots, E_n\}$ of n disks in the plane. For a disk set X, let $I(X)$ denote the intersection of the disks in X. If there is a bipartition B_ℓ such that both $I(\mathcal{E}_\ell)$ and $I(\mathcal{E}_\ell^c)$ are nonempty, the 2-piercing problem on $\mathcal{E}$ has the solution, where $\mathcal{E}_\ell$ is the set consisting of disks in $\mathcal{E}$ with centers lying strictly below ℓ, and $\mathcal{E}_\ell^c = \mathcal{E} \setminus \mathcal{E}_\ell$. For each bipartition B_ℓ, we perform the *emptiness test* which determines whether $I(\mathcal{E}_\ell) \neq \emptyset$ and $I(\mathcal{E}_\ell^c) \neq \emptyset$.

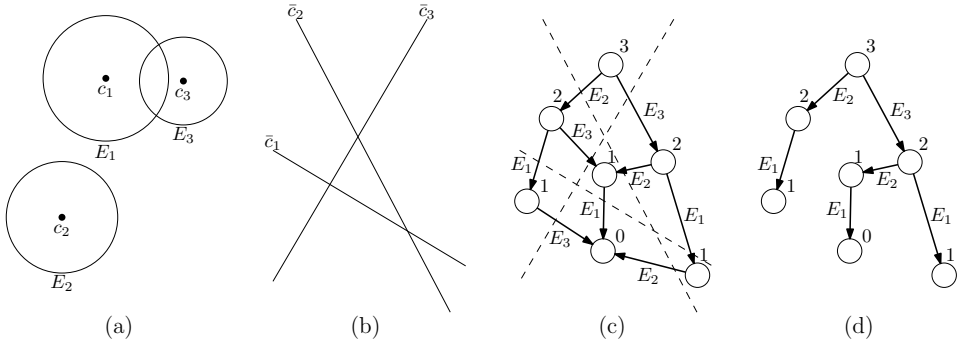


Figure 2 (a) Three disks E_1, E_2, E_3 with centers at c_1, c_2, c_3 in the plane. (b) Three dual lines $\bar{c}_1$, $\bar{c}_2$, $\bar{c}_3$ form the dual arrangement of the centers of disks in (a). (c) Dual graph $G_\mathcal{E}$ of the dual arrangement $\mathcal{A}$. The level of a node in $G_\mathcal{E}$ is the level of its corresponding face in $\mathcal{A}$. (d) Directed tree $T_\mathcal{E}$ from dual graph $G_\mathcal{E}$.

We construct the dual arrangement $\mathcal{A}$ for the centers of the disks in $\mathcal{E}$ by the following point-line duality transform: For a point $p := (p_x, p_y)$ in the primal plane, its dual $\bar{p}$ is the line $\bar{p} := (y = p_x x - p_y)$ in the dual plane. Likewise, for a line $\ell : y = \ell_x x + \ell_y$ in the primal plane, its dual $\bar{\ell}$ is the point $\bar{\ell} := (\ell_x, -\ell_y)$ in the dual plane. See Fig. 2(a,b). The duality transform preserves incidence ($p \in \ell$ if and only if $\bar{\ell} \in \bar{p}$) and order (p lies above ℓ if and only if $\bar{\ell}$ lies above $\bar{p}$) [6]. Thus, $\mathcal{A}$ is the arrangement induced by n lines in the dual plane, each of which is the dual of the center of an input disk. The level of a point in $\mathcal{A}$ is the number of lines in $\mathcal{A}$ lying on or below the point. For a face f of $\mathcal{A}$, let $\bar{\ell}$ be a point in f but not on the upper boundary chain of f. We define the level of f to be the level of $\bar{\ell}$. Let $\mathcal{E}_f$ denote the set of the disks in $\mathcal{E}$ such that the dual lines of their centers lie strictly above $\bar{\ell}$. Observe that $\mathcal{E}_f = \mathcal{E}_\ell$, and let $\mathcal{E}_f^c = \mathcal{E} \setminus \mathcal{E}_f$. Thus, they form the bipartition of the centers of input disks induced by ℓ in the primal plane.

3.1 Dual Directed Tree

Let $G_{\mathcal{E}}$ be a directed acyclic graph such that there is a one-to-one correspondence between the nodes of $G_{\mathcal{E}}$ and the faces in $\mathcal{A}$, and two nodes u, w of $G_{\mathcal{E}}$ are connected by a directed edge (u, w) from u to w if and only if the faces f_u and f_w corresponding to u and w, respectively, share a boundary edge and the level of f_u is larger than the level of f_w. There is a one-to-one correspondence between the bipartitions and the nodes of $G_{\mathcal{E}}$. For a node u in $G_{\mathcal{E}}$, let $\mathcal{E}_u = \mathcal{E}_f$ for face f of $\mathcal{A}$ corresponding to u, and let $\mathcal{E}_u^c = \mathcal{E} \setminus \mathcal{E}_u$. For each edge (u, w) of $G_{\mathcal{E}}$, $\mathcal{E}_w \setminus \mathcal{E}_u$ consists of exactly one disk, and (u, w) corresponds to the disk in $\mathcal{E}_w \setminus \mathcal{E}_u$. In Fig. 2(c), each directed edge is labeled with the disk corresponding to the edge.

Let v_r be the node of $G_{\mathcal{E}}$ that has no incoming edge. We construct from $G_{\mathcal{E}}$, a directed tree $T_{\mathcal{E}}$ rooted v_r that spans all vertices of $G_{\mathcal{E}}$, by choosing only one incoming edge for each node of $G_{\mathcal{E}}$. See Fig. 2(d). For two nodes u, w, let $p(u, w)$ denote the directed path from u to w in $T_{\mathcal{E}}$, if exists.

3.2 t-ary Search Trees

Let t be a parameter to be set later. For each leaf node v of $T_{\mathcal{E}}$, we construct a t-ary search tree $T_t(v)$ in bottom-up manner such that the leaf nodes are ordered from left to right, each corresponding to an edge in $p(v_r, v)$ in order from v_r to v, the leftmost t leaf nodes have the same parent node and the next t leaf nodes have the same parent node, and so on. This process goes recursively to higher levels, and $T_t(v)$ has height $h = O(\log_t n)$. See Fig. 3.

The path $p(v_r, v)$ represents a sequence of disks, each corresponding to an edge of the path. The data structure $T_t(v)$ supports queries that given a path $p(v_r, w)$ for a node w in $p(v_r, v)$, returns $h = O(\log_t n)$ subpaths that together form $p(v_r, w)$, and h intersections of disks, each corresponding to a subpath.

We construct a collection of t-ary search trees, one for each leaf node of $T_{\mathcal{E}}$, avoiding duplications of nodes as follows. First, we apply depth-first search (DFS) at v_r of $T_{\mathcal{E}}$, which gives us an order of the edges of $T_{\mathcal{E}}$, traversed by DFS. These edges are the leaf nodes of the collection, ordered from left to right following the order by DFS. Then we construct t-ary trees, in the order of the leaf nodes of $T_{\mathcal{E}}$ visited by DFS. For two leaf nodes v, v' with v visited before v', let v_{split} denote the lowest common ancestor node of $p(v_r, v)$ and $p(v_r, v')$. Then the path $p(v_r, v_{\text{split}})$ is the longest common subpath of the paths. To avoid duplications, $T_t(v')$ simply maintains a pointer to the part of $T_t(v)$ corresponding to $p(v_r, v_{\text{split}})$ with respect to t value, instead of constructing the part again. Let $\mathcal{T}_t$ denote the collection of all t-ary search trees. See Fig. 3(b) for an illustration.

3.3 Intersections of Disks for Paths

For a path $p(u, w)$, let $I(u, w)$ denote the intersection of the disks corresponding to $p(u, w)$. Observe that $I(v_r, w) = I(\mathcal{E}_w)$. For a node ν in $T_t(v)$, let ν^- be the left sibling node of ν, ν^+ the node next (right) to ν at the same level, and $\text{rc}(\nu)$ the rightmost child node of ν in $T_t(v)$. The leaf node ν of $T_t(v)$ corresponding to an edge e of $p(v_r, v)$ stores the intersection $I(\nu) = I(\nu^-) \cap D_e$ if ν^- is defined, and $I(\nu) = D_e$ otherwise, where D_e is the disk corresponding to e. A nonleaf node ν stores $I(\nu)$ if the subtree rooted at ν^+ is a perfect t-ary tree. We set $I(\nu) = I(\nu^-) \cap I(\text{rc}(\nu))$ if ν^- is defined, and $I(\nu) = I(\text{rc}(\nu))$ otherwise. See Fig. 3(b) for an illustration.

For a node ν, if $I(\nu)$ is stored at ν, $I(\nu) = I(w, w')$ for path $p(w, w')$ such that the edge of $p(w, w')$ incident to w corresponds to the leftmost leaf node in the subtree rooted at the

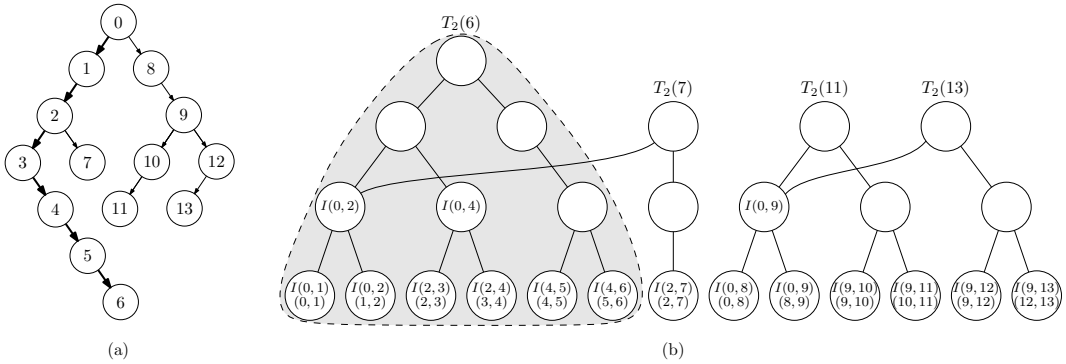


Figure 3 (a) Dual directed tree $T_{\mathcal{E}}$. (b) $\mathcal{T}_2$ for all leaf nodes in $T_{\mathcal{E}}$. $T_2(6)$ is constructed on $p(0,6)$ (thick path in (a)). Blank nodes in trees store no intersection of disks.

parent node of ν, and the edge of $p(w,w')$ incident to w' corresponds to the rightmost leaf node in the subtree rooted at ν. Thus, $I(\nu) = \bigcap D_e$ for all edges e in $p(w,w')$.

▶ **Lemma 3.1.** *Given a real value $r \geq 0$, we can construct $\mathcal{T}_t$ together with the intersections of disks stored at nodes in $O(tn^2 \log_t n)$ time using $O(tn^2 \log_t n)$ space.*

▶ **Lemma 3.2.** *Given a node w in $T_{\mathcal{E}}$, we can find a set $\mathcal{W}$ of $O(\log_t n)$ nodes in $\mathcal{T}_t$ such that $\cap_{\nu \in \mathcal{W}} I(\nu) = I(\mathcal{E}_w)$.*

3.4 Algorithm

If there is a node $u \in T_{\mathcal{E}}$ such that both $I(\mathcal{E}_u)$ and $I(\mathcal{E}_u^c)$ are nonempty, then the 2-piercing problem has a solution. Using $\mathcal{T}_t$ (Lemma 3.1 and Lemma 3.2), convex programming (Lemma 2.3) and setting $t = \log^{\epsilon} n$, we can solve the 2-piercing problem in $O(n^2 \log^2 n / \log\log n)$ time using $O(n^2 \log^{1+\epsilon} n)$ space for any constant $0 < \epsilon \leq 1$.

▶ **Theorem 3.3.** *Given a set of n disks in the plane, we can compute two points p_1 and p_2 such that every input disk contains p_1 or p_2 in $O(n^2 \log^2 n / \log\log n)$ time using $O(n^2 \log^{1+\epsilon} n)$ space for any constant $0 < \epsilon \leq 1$.*

4 The 2-center Problem on Disks

Our algorithms use parametric search which requires a sequential decision algorithm and a parallel decision algorithm.

4.1 Sequential Decision Algorithm

By solving the 2-piercing problem on the inflated disk in $\mathcal{D}(r)$, we can solve the decision 2-center problem with a given value r on $\mathcal{D}$.

▶ **Theorem 4.1.** *Given a set of n disks in the plane and a real value $r \geq 0$, we can determine whether there are two congruent disks C_1 and C_2 of radius r such that every input disk intersects C_1 or C_2 in $O(n^2 \log^2 n / \log\log n)$ time using $O(n^2 \log^{1+\epsilon} n)$ space for any constant ϵ with $0 < \epsilon \leq 1$.*

4.2 Parallel Decision Algorithm

We first describe a sequential preprocessing algorithm for finding an interval $(r_1, r_2]$ such that $r_1 < r^* \leq r_2$ and $\mathcal{T}_t$ has the same combinatorial structure for any $r \in (r_1, r_2]$, that is, for each intersection stored at nodes of $\mathcal{T}_t$, the circular arcs along the boundary are in the same order. The preprocessing consists of the construction of $\mathcal{T}_t$ for all $r \geq 0$ and binary search to find the interval $(r_1, r_2]$. To do this, we consider frustum F_i instead of $D_i(r)$ such that intersection of F_i and the plane $z = r$ is $D_i(r)$, for $i = 1, ..., n$.

▶ **Lemma 4.2.** *Given a set of n disks in the plane, we can construct $\mathcal{T}_t$ for all $r \geq 0$ in $O(tn^2 \log_t^2 n \cdot \log t)$ time. The space complexity of $\mathcal{T}_t$ for all $r \geq 0$ is $O(tn^2 \log_t n)$.*

▶ **Lemma 4.3.** *Given a set of n disks in the plane, we can find an interval $(r_1, r_2]$ in $O(n^2 \log^3 n / \log\log n)$ time such that $r_1 < r^* \leq r_2$ and $\mathcal{T}_t$ has the same combinatorial structure for any $r \in (r_1, r_2]$ and for $t = O(\log n)$.*

From the sequential preprocessing, we get $\mathcal{T}_t$ for an interval $(r_1, r_2]$ such that it has the same combinatorial structure for any $r \in (r_1, r_2]$ and $r^* \in (r_1, r_2]$. Using $\mathcal{T}_t$ and Lemma 2.2, we parallelize the process of determining $I(\mathcal{E}_u) = \emptyset$ and $I(\mathcal{E}_u^c) = \emptyset$ for all nodes $u \in T_{\mathcal{E}}$.

▶ **Theorem 4.4.** *Given a set of n disks in the plane and a real value $r \geq 0$, we can determine whether there are two congruent disks C_1 and C_2 of radius r such that every input disk intersects C_1 or C_2 in $O(\log n)$ time using $O(n^2 \log^2 n / \log^2 \log n)$ processors, after $O(n^2 \log^3 n / \log\log n)$-time preprocessing.*

4.3 Optimization Algorithms

We apply Cole's parametric search [5] to obtain an $O((P+T_s)(T_p+\log P))$-time deterministic algorithm, with our T_s-time sequential decision algorithm and our T_p-time parallel decision algorithm using P processors. Here $T_s = O(n^2 \log^2 n / \log\log n)$, $T_p = O(\log n)$ and $P = O(n^2 \log^2 n / \log^2 \log n)$. Thus, our deterministic algorithm runs in $O(n^2 \log^3 n / \log\log n)$ time. In addition, we apply Chan's randomized optimization [3] to obtain an $O(n^2 \log^2 n / \log\log n)$ expected time algorithm using our sequential decision algorithm.

▶ **Theorem 4.5.** *Given a set $\mathcal{D}$ of n disks in the plane, we can compute two smallest congruent disks C_1 and C_2 such that each disk in $\mathcal{D}$ intersects $C_1 \cup C_2$ in $O(n^2 \log^3 n / \log\log n)$ time. A randomized algorithm takes $O(n^2 \log^2 n / \log\log n)$ expected time.*

▶ **Corollary 4.6.** *Given a set of n disks in the plane, we can compute two smallest congruent disks C_1 and C_2 such that every disk is contained in either C_1 or C_2 in $O(n^2 \log^3 n / \log\log n)$ time. A randomized algorithm takes $O(n^2 \log^2 n / \log\log n)$ expected time.*

References

1 P. K Agarwal and M. Sharir. Planar geometric location problems. *Algorithmica*, 11(2):185–195, 1994.

2 H.-K. Ahn, S.-S. Kim, C. Knauer, L. Schlipf, C.-S. Shin, and A. Vigneron. Covering and piercing disks with two centers. *Computational Geometry*, 46(3):253–262, 2013.

3 T. M Chan. Geometric applications of a randomized optimization technique. *Discrete & Computational Geometry*, 22(4):547–567, 1999.

4 T. M Chan. More planar two-center algorithms. *Computational Geometry*, 13(3):189–198, 1999.

5 R. Cole. Slowing down sorting networks to obtain faster sorting algorithms. *Journal of the ACM (JACM)*, 34(1):200–208, 1987.

6 M. de Berg, O. Cheong, M. Van Kreveld, and M. Overmars. *Computational geometry algorithms and applications*. Springer, 3rd edition, 2008.

7 M. Sharir. A near-linear algorithm for the planar 2-center problem. *Discrete & Computational Geometry*, 18(2):125–134, 1997.

8 H. Wang. On the planar two-center problem and circular hulls. In *Proceedings of the 36th International Symposium on Computational Geometry*, volume 164, pages 68:1–68:14, 2020.

Online Ply Maintenance

Vikrant Ashvinkumar[1], Patrick Eades[2], Maarten Löffler[*3], and Seeun William Umboh[4]

1 **School of Computer Science, University of Sydney, Darlington, Australia**
2 **School of Computing and Information Systems, The University of Melbourne, Parkville, Australia**
3 **Department of Computing and Information Sciences, Utrecht University, the Netherlands**
4 **School of Computer Science, University of Sydney, Darlington, Australia**

Abstract

We introduce the *online ply maintenance problem*, and provide an online algorithm, a matching lower bound on the competitive ratio and a proof that perfect play is NP-hard, even with full information.

1 Introduction

1.1 The Ply of a Set of Regions

Given a set of regions $R = \{R_1, \ldots, R_n\}$ in $\mathbb{R}^d$, the *local ply* of a point $p \in \mathbb{R}^d$ in R is defined $\delta_p = |\{R_i \mid p \in R_i\}|$, i.e. the number of regions containing that point. The *ply* of R is $\delta = \max_{p \in \mathbb{R}^d} \delta_p$ the maximum local ply of any point.

The ply of a set of *imprecise points*[1] gives a measure of how much is unknown about the geometry of the set of points. If the ply can be upper bounded by some constant Δ then many algorithms become easier, often with running times depending on Δ. For example, Buchin *et al.* [1] show how a data structure can be built on a set of n disks of ply Δ, such that given a point in each disk a Delaunay triangulation can be found in $\mathcal{O}(n \log \Delta)$.

1.2 Problem Statement

We define our problem as a game played by a player and by an adversary over continuous, unbounded time. Let $e_1, \ldots, e_n$ be a set of entities moving in $\mathbb{R}^d$ where each e_i is associated with a fixed (but unknown to the player) trajectory $f_i : \mathbb{R}_{\geq 0} \to \mathbb{R}^d$ with a maximum speed of 1 per unit of time. The initial location $f_i(0)$ of each trajectory is known to the player. The adversary meanwhile has full knowledge of each f_i and has unlimited computation.

At any time t the player may query an entity e_i to (instantaneously) learn its exact location $f_i(t)$. Let τ be the last time at which e_i was queried, or 0 if e_i has never been queried. At time t we can infer e_i must be located in a closed ball of radius $t - \tau$ centered at $f_i(\tau)$. Call this ball $B_i(t)$ the *uncertainty region* of e_i at time t.

We can now formulate the *online ply maintenance problem*. Given a ply bound Δ and a starting location $f_i(0)$ and location oracle f_i for each entity, the online ply maintenance problem is to determine which points the player should query at which times, such that the ply of the uncertainty regions never exceeds Δ and the total number of queries is minimized. For simplicity we require that all problem instances are feasible; that is, there is never a time when more than Δ of the entities are present at the same point.

* Partially supported by the Dutch Research Council (NWO); 614.001.504.

[1] Points given by a set of regions, each region contains a point but its exact position is unknown; e.g. [9]

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

This is an online problem; the player may use information gained from past queries to decide which queries to make next. There is also no finishing time for the problem, the player must provide a strategy that works for any time and the number of queries must be minimized at all times.

Since a large number of queries may be required regardless of strategy, the player's performance is evaluated against the adversary, who plays a copy of the same game except with full knowledge of the f_i. The adversary must still contend with growing uncertainty regions and must make queries to maintain low ply, but knows the result of every query beforehand and may plan accordingly.

Define $\text{cost}(t)$ as the number of queries required by an algorithm for the player by time t and $\text{cost}^*(t)$ the number of queries required by the adversary. We say the algorithm has competitive ratio α if $\text{cost}(t) \leq \alpha \cdot \text{cost}^*(t)$ for all $t > 0$.

For simplicity of exposition, we will allow the ply to exceed Δ instantaneously, so long as queries are made at the same instant which return the ply to Δ or less. This allows queries to be made *at* events in time rather than ϵ time before them. We also observe that these are the only kinds of queries made by any efficient algorithm.

▶ Observation 1. Any query made on an entity at a time when it is not contributing to the ply can be replaced with a query on the same entity at a later time when it is contributing to the ply, without requiring that any additional queries are made.

As a corollary, any efficient algorithm will wait until the ply instantaneously exceeds Δ and make a sequence of queries to reduce the ply, then wait until the ply exceeds Δ again. Thus the continuous problem is best understood as a series of events and the queries which are made at each event, which is how we will discuss the problem from now on. It is worth noting that two different algorithms will not necessarily have the same events.

1.3 Prior Work

This problem statement is inspired by Evans *et al.* [5], who study a similar problem. Their problem has the same setting, except only one query can be made per unit of time. Instead of minimizing the number of queries, the ply must be minimized. It is not feasible to maintain optimal ply at all times, hence Evans *et al.* must fix a target time τ and minimize the ply *at that time*, once or repeating every τ time interval. Evans *et al.* give an $\mathcal{O}(1)$-approximation for large τ, an approximation for small τ, and matching lower bounds. They also show the adversary's problem is NP-hard. Busto *et al.* [2] give an extension. Since Evans *et al.* demonstrated it is not possible to be competitive at *every* time step, Busto *et al.* instead consider minimizing the maximum over a given time interval. They provide an algorithm which is competitive to within a constant factor.

Our result is a natural pair to Evans *et al.* and Busto *et al.*. We are able to guarantee a maximum ply (and thus bound the performance of any subsequent ply-dependent algorithm), and our guarantee holds at *any* (continuous) point in time, not just at fixed time intervals or over an interval. On the other hand we may use many more queries.

Acknowledgement must be made to the work of Kahan [7, 8]. Kahan introduces a model called *data in motion* which is similar to our own, although it is aimed at a different application. Kahan considers the time and number of queries required to compute some geometric property (e.g. the closest pair) of the entities at various points in time.

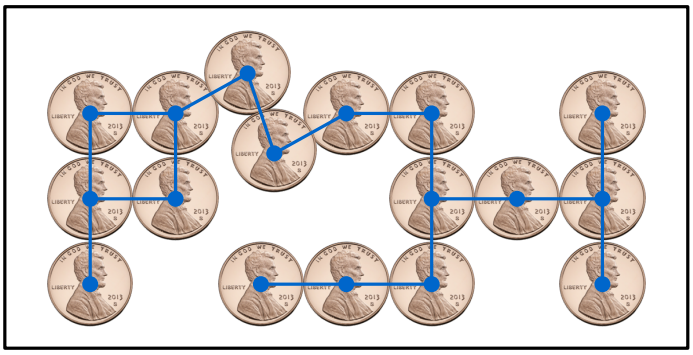


Figure 1 Some pennies arranged along a grid and the resulting penny graph. Note the slight divergence from the grid, which is sometimes required by Cerioli *et al.* for parity reasons.

2 Optimal Play is NP-hard

In this section we will show that the adversary's problem of deciding which entities to query is NP-hard, even given full knowledge of the trajectories beforehand. We only need to consider the problem of deciding which queries to make at the very first time the ply exceeds $\Delta = 1$ to see this behaviour. We will perform a reduction from vertex cover.

▶ **Lemma 2.1.** *Given a set of n equal-radius disks in the plane, possibly intersecting at tangent points but not overlapping, the problem of determining if it is possible to remove k disks such that the remaining disks are pairwise disjoint is NP-complete.*

A *penny graph*, sometimes called a *unit coin graph*, is a graph which can be realized as the intersection graph of a set of unit disks in the plane which only intersect at tangents.

Cerioli *et al.* [3] show that vertex cover is NP-hard, even on penny graphs. However our penny graph will be specified by disk centres, not edges and vertices. A graph representation can be easily computed from the disk centres [11], but the converse is not true [4].

Vertex cover is NP-hard even for planar graphs of degree no more than 3 [6], and such graphs can be embedded on a polynomial-sized grid with rectilinear edges [10]. Given such a graph G, embedded onto a grid, Cerioli *et al.* show how pennies can be laid out such that solving vertex cover on the resulting penny graph solves vertex cover on G. Hence vertex cover on penny graphs is NP-hard even if the disk centres are known.

Choosing which disks to remove such that no disks intersect is exactly the problem of computing a vertex cover of the intersection graph, hence Lemma 2.1 follows. Given a instance of the unit disk-removing problem we can construct an instance of ply maintenance with a stationary entity at the center of each disk. After one unit of time the disks will intersect at tangents, and the adversary must decide a minimum number of disks to query to remove the intersections. By Lemma 2.1 even this first decision is NP-hard.

3 An Optimal Algorithm

We now describe a simple algorithm which achieves a competitive ratio of $\Delta + 1$, which we will later show to be optimal for deterministic algorithms. Our algorithm waits until there is a point with ply at least $\Delta + 1$ and then queries *every* entity whose uncertainty region contains that point. If there are multiple points with ply at least $\Delta + 1$, say p and q, they are resolved in sequence; if querying all the entities intersecting p reduces the ply of q to Δ or less, then it is not necessary to query all the entities intersecting q.

We will prove our algorithm has a competitive ratio of $\Delta + 1$ using a charging argument. We charge every query we make to a query made by the adversary, and ensure we never charge more than $\Delta + 1$ of our queries to a single query of the adversary.

Consider a time when our algorithm has ply $\Lambda > \Delta$. If the adversary must make any queries at the same time we imagine for simplicity that the adversary makes them first and then our algorithm makes its queries second.

Consider the first point of ply Λ that our algorithm must resolve. At least $\Lambda - \Delta$ of the uncertainty regions that contribute to the ply of this point must be strictly smaller for the adversary than they are for our algorithm. Otherwise the point would have ply more than Δ for the adversary, and since the adversary has already made its queries it cannot have any point of ply more than Δ. We will call this set of queries made by the adversary Q.

We charge the Λ queries our algorithm makes to adversary's queries Q. By simple algebra $\Lambda/(\Lambda - \Delta) \leq \Delta + 1$ and so no query in Q has more than $\Delta + 1$ queries charged to it.

Observe any query the adversary makes can only be charged to once. A query can only be charged to by a later query on an entity where the adversary has a strictly larger uncertainty region than our algorithm. When we query an entity its uncertainty region is reset to a single point, hence it will have a smaller or equal uncertainty region to the adversary. Only when the adversary queries that entity again will it be eligible to receive a charge, but then there will be a new query made by the adversary to pay for the charge.

Since we can always charge our algorithm's prior queries to an earlier query made by the adversary, and we have ensured there is no over-charging, we can be sure that at no time has our algorithm made more than $\Delta + 1$ times the number of queries made by the adversary. Since our algorithm trivially always maintains ply, the theorem follows.

▶ **Theorem 3.1.** *Our algorithm is* $(\Delta + 1)$*-competitive.*

4 Lower Bound

In this section we give a lower bound of 2 for $\Delta = 1$, matching the competitive ratio given in the previous section. The lower bound may be extended to $\Delta + 1$ for $\Delta \geq 1$ but this has been omitted due to space constraints.

Consider an instance in $\mathbb{R}$ with two entities, p and q, initially located at -1 and 1. The instance includes an arbitrary sequence of moves $m_1, m_2, \ldots \in \{\text{left}, \text{right}\}$, known to the adversary but not the player. Between $t = i$ and $t = i + 1$ both entities move at unit speed in the direction m_i. Because the entities are always distance 2 apart, queries will be needed at least once per unit of time. We will show in the worst case the player must make exactly two queries per unit of time, while the adversary must only make one.

We first consider the player. At $t = 1$ the uncertainty regions will intersect and the player must make a query. By Observation 1 it is never beneficial to query an entity early, so the player will not make any query before $t = 1$. Imagine both entities have moved left. The player has no way to distinguish the entities and must query one arbitrarily, in the worst case they will choose q. Since q started at 1 and moved left it is now at 0, inside the uncertainty region of p, so the ply remains 2. Hence the player must also query p to reduce the ply. Now each entity is represented by a single point, at distance 2 apart. This is the same situation we began with (shifted sideways) and so the same argument can be repeated at $t = 2$ and so on. Therefore, in the worst case, the player must make two queries per unit time.

We now show the adversary must only make one query per unit of time. The adversary will not be querying every point each unit of time, so uncertainty regions may grow arbitrarily large. We will say the problem has a neutral state at some time if one entity is represented

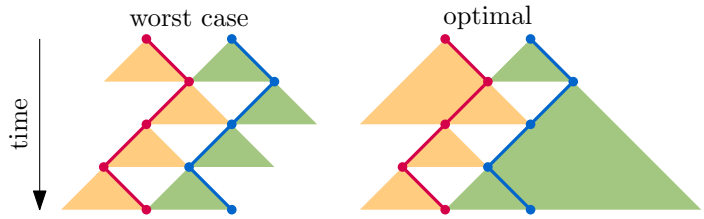


Figure 2 In the worst case the player needs two queries per unit of time, while the adversary never makes a wasted query and only ever needs one.

by a single point and the other uncertainty region is at distance 2. For illustration say p is represented by a point at 1 and the uncertainty region for q extends from -1 in the negative direction. Clearly the state at $t = 0$ is neutral. Unit time after a neutral state the two uncertainty regions will now intersect at 0 and a query must be made. The entity p is now either at 0 or at 2. If it is at 2 then the adversary queries p and reduces it to a single point at 2, since the uncertainty region for q ends at 0 this is a neutral state. If p is at 0 then by construction q must be at -2. The adversary queries q, since the uncertainty region for p is $[0, 2]$ this is a neutral state. Therefore one unit of time after a neutral state the adversary can spend one query to return to a neutral state, and so the adversary need only make one query per unit of time. Together these statements give a lower bound on the competitive ratio of any algorithm of 2, for the case when $\Delta = 1$. See Figure 2 for an illustration.

Finally consider allowing the player to make a randomized decision of which entity to query. By Yao's principle [12] we may instead consider the best expected cost of any deterministic algorithm on a random instance. We define our random instance as above, except the sequence of left and right moves is chosen uniformly at random. As above, any deterministic algorithm has no information to distinguish the entities and must arbitrarily choose one to query. Half the time it will choose correctly, the other half of the time it must immediately make a second query. Therefore on average the algorithm must make 1.5 queries per unit of time. By Yao's principle this is a lower bound on the competitive ratio for any randomized algorithm. A similar argument gives a lower bound of $\frac{1}{2}\Delta + 1$ in the general case.

References

1 Kevin Buchin, Maarten Löffler, Pat Morin, and Wolfgang Mulzer. Preprocessing imprecise points for delaunay triangulation: Simplified and extended. *Algorithmica*, 61(3):674–693, 2011. doi:10.1007/s00453-010-9430-0.

2 Daniel Busto, William S. Evans, and David G. Kirkpatrick. Minimizing interference potential among moving entities. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2400–2418. SIAM, 2019. doi:10.1137/1.9781611975482.147.

3 Márcia R Cerioli, Luerbio Faria, Talita O Ferreira, and Fábio Protti. On minimum clique partition and maximum independent set on unit disk graphs and penny graphs: complexity and approximation. *Electronic Notes in Discrete Mathematics*, 18:73–79, 2004.

4 Peter Eades and Sue Whitesides. The logic engine and the realization problem for nearest neighbor graphs. *Theor. Comput. Sci.*, 169(1):23–37, 1996. doi:10.1016/S0304-3975(97)84223-5.

5 William S. Evans, David G. Kirkpatrick, Maarten Löffler, and Frank Staals. Competitive query strategies for minimising the ply of the potential locations of moving points. In

Guilherme Dias da Fonseca, Thomas Lewiner, Luis Mariano Peñaranda, Timothy M. Chan, and Rolf Klein, editors, *Symposuim on Computational Geometry 2013, SoCG '13, Rio de Janeiro, Brazil, June 17-20, 2013*, pages 155–164. ACM, 2013. doi:10.1145/2462356.2462395.

6 Michael R Garey and David S. Johnson. The rectilinear steiner tree problem is np-complete. *SIAM Journal on Applied Mathematics*, 32(4):826–834, 1977.

7 Simon Kahan. A model for data in motion. In Cris Koutsougeras and Jeffrey Scott Vitter, editors, *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 267–277. ACM, 1991. doi:10.1145/103418.103449.

8 Simon H. Kahan. *Real-time processing of moving data.* PhD thesis, University of Washington, 1991.

9 Maarten Löffler and Marc J. van Kreveld. Largest and smallest convex hulls for imprecise points. *Algorithmica*, 56(2):235–269, 2010. doi:10.1007/s00453-008-9174-2.

10 Leslie G Valiant. Universality considerations in vlsi circuits. *IEEE Transactions on Computers*, 100(2):135–140, 1981.

11 Remco Veltkamp. *Closed object boundaries from scattered points*, volume 885. Springer Science & Business Media, 1994.

12 Andrew Chi-Chih Yao. Probabilistic computations: Toward a unified measure of complexity (extended abstract). In *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977*, pages 222–227. IEEE Computer Society, 1977. doi:10.1109/SFCS.1977.24.

Hypergraph Representation via Axis-Aligned Point-Subspace Cover

Oksana Firman[1] and Joachim Spoerhase[1]

1 **Universität Würzburg, Germany**
`firstname.lastname@uni-wuerzburg.de`

Abstract

We propose a new representation of k-partite, k-uniform hypergraphs (which we call k-hypergraphs for short) by a finite set P of points in $\mathbb{R}^d$ and a parameter $\ell \leq d-1$. Each point in P is covered by $k = \binom{d}{\ell}$ many axis-aligned affine ℓ-dimensional subspaces of $\mathbb{R}^d$, which we call ℓ-subspaces for brevity. We interpret each point in P as a hyperedge that contains each of the covering ℓ-subspaces as a vertex. The class of (d,ℓ)-*hypergraphs* is the class of k-hypergraphs that can be represented in this way, where $k = \binom{d}{\ell}$. The resulting classes of hypergraphs are fairly rich: Every k-hypergraph is a $(k, k-1)$-hypergraph. On the other hand, (d,ℓ)-hypergraphs form a proper subclass of the class of all $\binom{d}{\ell}$-hypergraphs for $\ell < d-1$.

We are able to give a natural structural characterization of (d,ℓ)-hypergraphs based on vertex cuts. This characterization leads to a polynomial-time recognition algorithm that decides for a given $\binom{d}{\ell}$-hypergraph whether or not it is a (d,ℓ)-hypergraph and that computes a representation if existing. We assume that the dimension d is constant and that the partitioning of the vertex set is prescribed. For the sake of presentation, we describe in this paper our result for the case of $(d,1)$-hypergraphs, that is, for covering points with axis-parallel lines. This special case can be naturally extended to the general case.

1 Introduction

Motivation and Related Work. Geometric representations of graphs or hypergraphs is wide and intensively studied field of research. Well-known examples are geometric intersection or incidence graphs with a large body of literature [12, 5, 13]. The benefit of studying geometric graph representations is two-fold. On the one hand, knowing that a given graph can be represented geometrically may give new insights because the geometric perspective is often more intuitive. On the other hand, giving a graphical characterization for certain types of geometric objects may help pin down the essential combinatorial properties that can be exploited in the geometric setting.

One example of this interplay is the study of geometric set cover and geometric hitting set problems in geometric optimization [2, 14, 3]. In this important branch of computational geometry, incidence relations of two types of geometric objects are studied where one object type is represented by vertices of a hypergraph whose hyperedges are, in turn, represented by the other object type. In this representation a vertex is contained in a hyperedge if and only if the corresponding geometric objects have a certain geometric relation such as containment or intersection. The objective is to find the minimum number of nodes hitting all hyperedges[1]. In this line of research, the goal is to exploit the geometry in order to improve upon the state of the art for general hypergraphs. This is known to be surprisingly challenging even in many seemingly elementary settings.

[1] For the sake of presentation, we use here the representation as hitting set problem rather than the equivalent and maybe more common geometric set cover interpretation.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

For example, in the well-studied point line cover problem [8, 10] we are given a set of points in the plane and a set of lines. The goal is to identify a smallest subset of the lines to cover all the points. This problem can be cast as a hypergraph vertex cover problem. Points can be viewed as hyperedges containing the incident lines as vertices. The objective is to cover all the hyperedges by the smallest number of vertices.

It seems quite clear that point line cover instances form a heavily restricted subclass of general hypergraph vertex cover. For example, they have the natural intersection property that two lines can intersect in at most one point. However, somewhat surprisingly, in terms of approximation algorithms, no worst-case result improving the ratios for general hypergraph vertex cover [1, 7, 4] is known. In fact, it has been shown that merely exploiting the above intersection property in the hypergraph vertex cover is not sufficient to give improved approximations [11]. Giving a simple combinatorial characterization of the point line cover instances seems to be an intriguing task.

In this paper, we study a representation of hypergraphs that arises from a natural variant of point line cover where we want to cover a given set P of points in $\mathbb{R}^d$ by *axis-parallel* lines. While the axis-parallel case of point line cover has been considered before [6] the known algorithms do not improve upon the general case of hypergraph vertex cover [1, 4]. More generally, we investigate the generalization where we are additionally given a parameter $\ell \leq d-1$ and we would like to cover P by axis-aligned affine ℓ-dimensional subspaces of $\mathbb{R}^d$, which we call ℓ-subspaces. The resulting classes of hypergraphs is fairly rich as any k-partite k-uniform hypergraph (i.e. a hypergraph with a partition of vertices into k parts such that each hyperedge contains at most one vertex of each type and together exactly k vertices) can be represented by a set of points in $\mathbb{R}^k$ to be covered by $(k-1)$-subspaces. On the other hand for $\ell < d-1$ we obtain proper subclasses of all k-hypergraphs.

We remark that our representation does not exploit the geometry of the Euclidean $\mathbb{R}^d$. Rather, the representation can also be considered on a hypercube X^d for some set X where subspaces are subsets of X^d fixing certain coordinates. We feel that the usage of the geometric language is more intuitive.

Our Contribution To the best of our knowledge, we are the first to study the representation of k-hypergraphs via axis-aligned point subspace cover instance in this generality. Our main insight is that the axis-aligned case of point subspace cover allows for a natural, combinatorial characterization contrasting what is known for the non-aligned case. The characterization is based on vertex cuts and can be leveraged to obtain a polynomial time recognition algorithm for such hypergraphs assuming the dimension d is a constant and that we are given the partition of the vertices (which is NP-hard to compute for $k \geq 3$ [9]). We believe that it is an interesting research direction to exploit these combinatorial properties in order to obtain improved results for various optimization problems in (d, ℓ)-hypergraphs such as hypergraph vertex cover or hypergraph matching.

2 Point Line Cover and Hypergraph Representation

For the sake of an easier presentation, we describe the result for the special case of point line covers. We give later some intuition how to generalize the results to high-dimensional axis-aligned affine subspaces.

Let P be a finite set of points in $\mathbb{R}^d$. Then we define the hypergraph G_P as follows. The vertex set of G_P is the set of axis-parallel lines containing at least one point in P. The hyperedges in G_P correspond to the points in P where the hyperedge corresponding to some

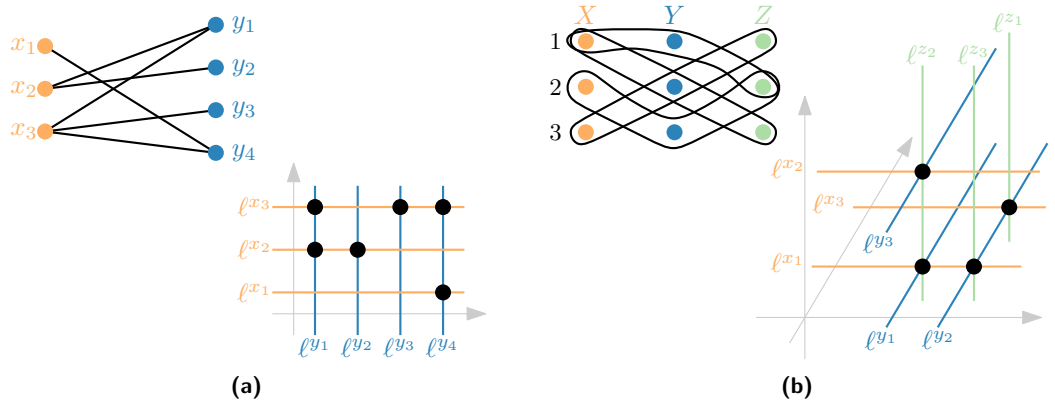


Figure 1 A graph (a) and a hypergraph (b) and their representations in 2D and 3D, respectively.

$p \in P$ contains the k axis-parallel lines incident on p as vertices. Note that G_P is k-partite and k-uniform (that is G_P is a k-hypergraph) where the k groups of the partition correspond to the k dimensions.

Our main task is to decide for a given hypergraph G whether there is a point line cover instance P such that G and G_P are isomorphic. We say that G is *represented* by P and, thus, *representable*. We assume that the partition of G into k groups is given.

More formally, we want to compute for a given k-hypergraph $G = (V_1 \cup V_2 \cup, \ldots, \cup V_k, E)$ a point line cover instance P such that each $e = (v_1, \ldots, v_k) \in E$ corresponds to some $p^e = (x_1^e, \ldots, x_k^e) \in P$ and where $v_i \in V_i$ corresponds to the line l^{v_i} that is parallel to the ith coordinate axis and contains p_e, that is, for all $j \neq i$, we fix the coordinates $x_j^e, j \neq i$ whereas the ith coordinate is free, see Fig. 1 for examples.

We remark that every bipartite graph is representable in $\mathbb{R}^2$ because any adjacency matrix can be represented by a grid-like construction as shown in Fig. 1a. Therefore, from now on we consider the case $k \geq 3$.

3 Characterization of Representable Hypergraphs

We use the notation $[k] = \{1, \ldots, k\}$ for $k \in \mathbb{N}$. Let $G = (V = V_1 \cup \cdots \cup V_k, E)$ be a k-hypergraph.

▶ **Definition 3.1.** Let $s, t \in V$. An *s–t path* is a sequence of vertices $s = v_1, \ldots, v_r = t$ such that v_i and v_{i+1} are both contained in some hyperedge $e \in E$ for all $i \in [r-1]$. Similarly, if $e, e' \in E$ then an *e–e' path* is a v–v' path such that $v \in e$ and $v' \in e'$.

▶ **Definition 3.2** (Vertex separability)**.** For a given k-hypergraph G two distinct vertices v and v' from the same group V_i, $i \in [k]$ are *separable* if there exists some $j \in [k]$ with $j \neq i$ such that every v–v' path contains a vertex in V_j. (Informally, removing V_j from the vertex set and from the edges separates v and v'.) A hypergraph is called *vertex-separable* if every two vertices from the same group are separable.

▶ **Definition 3.3** (Edge separability)**.** For a given k-hypergraph G two distinct hyperedges e and e' are *separable* if there exists some $j \in [k]$ such that every e–e' path contains a vertex in V_j. A hypergraph is called *edge separable* if every two hyperedges are separable.

Note that any pair of hyperedges sharing two or more vertices are not separable. Therefore, edge-separable hypergraphs do not contain such hyperedge pairs.

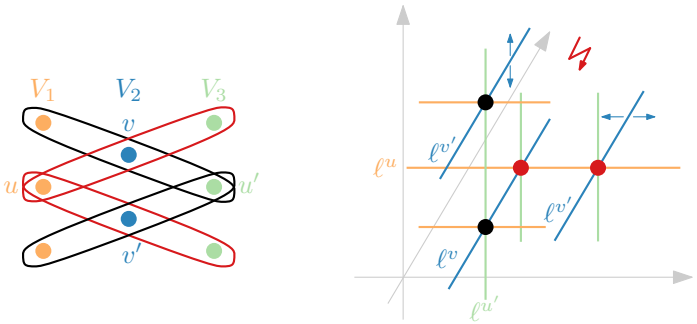


Figure 2 A hypergraph (on the left) that is edge-separable, but not vertex-separable (the vertices from V_2 are not separable). The line $\ell^{v'}$ (on the right) must simultaneously intersect ℓ^u and $\ell^{u'}$ and therefore must be equal to ℓ^v. A contradiction.

▶ **Lemma 3.4.** *Vertex separability implies edge separability.*

Proof. Assume that a given k-hypergraph G is not edge-separable. This means that there are two distinct hyperedges e and e' that are not separable. Then $\forall j \in [k]$ there is an e–e' path that does not contain a vertex from V_j. Because e and e' are distinct there are distinct vertices v and v' with $v \in e$ and $v' \in e'$ from the same group V_i for some $i \in [k]$. Now, for each $j \in [k]$, $j \neq i$ there exists an e–e' path P_j that does not contain any vertex from V_j. But then v, P_j, v' forms a v–v' path not containing any vertex from V_j. This means that G is not vertex-separable. ◀

The converse is not true, see Fig. 2. In the instance depicted, the two red edges, for example, are separated by removing the orange vertex part V_1 and the two black edges are separated by removing the green vertex part V_3.

▶ **Definition 3.5.** Let G be a k-hypergraph. For each $i \in [k]$ we construct a graph $G_i = (E, E_i)$ as follows: e and $e' \in E$ are adjacent if and only if e and e' have a common vertex in a group V_j with $j \neq i$.

▶ **Theorem 3.6.** *A k-hypergraph G is representable if and only if it is vertex-separable.*

Proof. We construct for each hyperedge e a point $p^e \in \mathbb{R}^k$ and for each vertex $v_i \in V_i$ with $i \in [k]$ a line $l^{v_i} \subseteq \mathbb{R}^k$ that is parallel to the x_i-axis. We do this as follows. For G we construct the graphs G_i, $i \in [k]$. For each graph G_i we consider the connected components of the graph and assign to each of them a unique (integer) value.

Now, if p^e_i is the value of the connected component in G_i that contains e then we let the point $p^e = (p^e_1, \ldots, p^e_k)$ represent the hyperedge e, see Fig. 3 for an example.

Recall that any line parallel to the x_i-axis can be defined by fixing its x_j-coordinate for all $j \neq i$, while leaving x_i free. Now, if the hyperedge $e = \{v_1, \ldots, v_k\}$ is represented by $p^e = (p^e_1, \ldots, p^e_k)$ then for each $i \in [k]$, the line l^{v_i} that represents the vertex v_i is defined by coordinates p^e_j, $j \neq i$ while leaving the x_i-coordinate free, see Fig. 3. It is important to note, that the representation l^{v_i} is well-defined although v_i may be contained in multiple hyperedges in G. This follows from the fact that all the hyperedges containing v_i belong to the same connected component in G_j, $j \in [k]$, $j \neq i$ because each pair of them is joined by some edge in G_j corresponding to v_i and in particular these hyperedges form a clique. Therefore, there is no disagreement in the x_j-coordinate where $j \neq i$. Hence, we uniquely define the coordinates that determine a line.

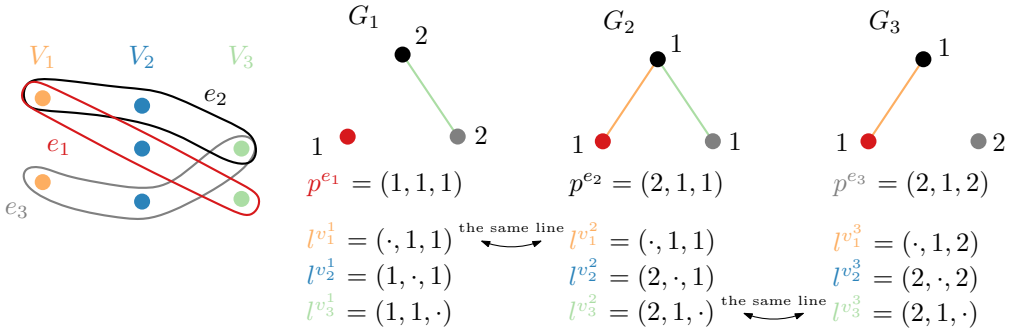


Figure 3 The graphs G_1, G_2, G_3 and the coordinates of the points and lines corresponding to the hyperedges and vertices. The dots instead of coordinates mean that those coordinates are free.

($\Leftarrow$) Assume that G is vertex-separable. By the construction of the point line cover instance we have:

- every point p^e is in fact covered by the lines $l^{v_1}, \ldots, l^{v_k}$ where $e = \{v_1, \ldots, v_k\}$, because by construction every line l^{v_i} and point p^e have the same x_j-coordinate with $j \neq i$.
- $\forall v \neq v' \in V$ it holds $l^v \neq l^{v'}$. This is obviously true if vertices belong to different groups, because then the free coordinate of v is fixed for v' and vice versa. If $v, v' \in V_i$ for some $i \in [k]$ then, by vertex separability, there exists $j \neq i$ such that v and v' are not connected in graph G_j and get different x_j-coordinates. So they represent distinct lines.
- $\forall e \neq e' \in E$ it holds $p^e \neq p^{e'}$. Indeed, by Lemma 3.4, G is edge-separable and by definition of edge separability distinct hyperedges are not connected in at least one graph G_i and get different x_i-coordinates. So they represent distinct points.

By the above construction, for every incident vertex-hyperedge pair $v \in V, e \in E$, that is, $v \in e$, the corresponding geometric objects l^v and p^e are incident as well. We claim that if v and e are not incident, that is, $v \notin e$ then l^v and p^e are not incident as well. This is because every point p^e is already incident to precisely k lines l^v by construction, because the lines l^v are pairwise distinct, and because p_e cannot be incident on more than k axis-parallel lines. Thus we constructed a point line cover instance that represents the hypergraph G and this means that G is representable.

($\Rightarrow$) Assume that G is not vertex-separable but that it has a point line cover representation. This means that it contains at least two distinct vertices v and v' from the same group V_i that are not separable. Then for each group V_j with $j \neq i$, there exists a v–v' path $v = v_1, \ldots, v_r = v'$ such that $v_t \notin V_j$ for each $t \in [r]$. All lines l^{v_t} with $t \in [r]$ that represent the vertices $v_1, \ldots, v_r$ lie on the same hyperplane H_j perpendicular to the x_j-axis. This is because successive line pairs are joined by a common point (representing the hyperedge containing both) and since none of these lines is parallel to the x_j-axis and so the x_j-coordinate stays fixed. Since this holds for all $j \in [k]$, $j \neq i$, the lines l^v and $l^{v'}$ lie in the intersection $\bigcap_{j \neq i} H_j$. But the intersection of such hyperplanes is a single line. This contradicts that v and v' correspond to the distinct lines. ◀

4 Further Results and Open Questions.

In the full version of the paper, we also present a polynomial time algorithm to compute for a vertex separable hypergraph a point line cover representation. The algorithm implements the idea used in Theorem 3.6. We are also able to generalize the result from point line covers

to point subspace covers. That is, we provide a characterization of the more general case of (d, ℓ)-hypergraphs along with a recognition algorithm for constant d.

We conclude with some open questions. Can we leverage our combinatorial characterizations to give improved algorithms for classical optimization problems such as hypergraph vertex cover or hypergraph matching for (d, ℓ)-hypergraphs? What is the relation of such graphs to other more well-studied graph classes? Is there a polynomial recognition algorithm also for non-constant d? What about point line cover in the plane with a fixed number of possible directions of the lines?

References

1 Ron Aharoni, Ron Holzman, and Michael Krivelevich. On a theorem of lovász on covers in tau-partite hypergraphs. *Comb.*, 16(2):149–174, 1996. doi:10.1007/BF01844843.

2 Hervé Brönnimann and Michael T. Goodrich. Almost optimal set covers in finite vc-dimension. *Discret. Comput. Geom.*, 14(4):463–479, 1995. doi:10.1007/BF02570718.

3 Timothy M. Chan, Elyot Grant, Jochen Könemann, and Malcolm Sharpe. Weighted capacitated, priority, and geometric set cover via improved quasi-uniform sampling. In Yuval Rabani, editor, *Proc. Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'12)*, pages 1576–1585. SIAM, 2012. doi:10.1137/1.9781611973099.125.

4 Vasek Chvátal. A greedy heuristic for the set-covering problem. *Math. Oper. Res.*, 4(3):233–235, 1979. doi:10.1287/moor.4.3.233.

5 H. S. M. Coxeter. Self-dual configurations and regular graphs. *Bull. Amer. Math. Soc.*, 56(5):413–455, 09 1950.

6 Daya Ram Gaur and Binay Bhattacharya. Covering points by axis parallel lines. In *Proc. 23rd European Workshop on Computational Geometry (EuroCG'07)*, pages 42–45. Citeseer, 2007.

7 Venkatesan Guruswami and Rishi Saket. On the inapproximability of vertex cover on k-partite k-uniform hypergraphs. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *Proc. 37th International Colloquium on Automata, Languages and Programming (ICALP'10)*, volume 6198 of *Lecture Notes in Computer Science*, pages 360–371. Springer, 2010. doi:10.1007/978-3-642-14165-2_31.

8 Refael Hassin and Nimrod Megiddo. Approximation algorithms for hitting objects with straight lines. *Discret. Appl. Math.*, 30(1):29–42, 1991. doi:10.1016/0166-218X(91)90011-K.

9 Lucian Ilie, Roberto Solis-Oba, and Sheng Yu. Reducing the size of NFAs by using equivalences and preorders. In Alberto Apostolico, Maxime Crochemore, and Kunsoo Park, editors, *Combinatorial Pattern Matching*, pages 310–321. SV, 2005. doi:10.1007/11496656_27.

10 Stefan Kratsch, Geevarghese Philip, and Saurabh Ray. Point line cover: The easy kernel is essentially tight. In Chandra Chekuri, editor, *Proc. 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'14)*, pages 1596–1606. SIAM, 2014. doi:10.1137/1.9781611973402.116.

11 V. S. Anil Kumar, Sunil Arya, and H. Ramesh. Hardness of set cover with intersection 1. In Ugo Montanari, José D. P. Rolim, and Emo Welzl, editors, *Proc. International Colloquium on Automata, Languages and Programming (ICALP'00)*, volume 1853 of *Lecture Notes in Computer Science*, pages 624–635. Springer, 2000. doi:10.1007/3-540-45022-X_53.

12 Terry A. McKee and F. R. McMorris. *Topics in Intersection Graph Theory*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, 1999.

13 Tomaž Pisanski and Milan Randić. Bridges between geometry and graph theory. In *in Geometry at Work, C.A. Gorini, ed., MAA Notes 53*, pages 174–194. America.

14 Kasturi R. Varadarajan. Weighted geometric set cover via quasi-uniform sampling. In Leonard J. Schulman, editor, *Proc. 42nd ACM Symposium on Theory of Computing (STOC'10)*, pages 641–648. ACM, 2010. doi:10.1145/1806689.1806777.

Minimizing the Maximum Interference in Dual Power Sensor Networks

Aviad Baron

Department of Computer Science, Ben Gurion University, Beer-Sheva 84105, Israel
baronav@post.bgu.ac.il

Abstract

Let P be a set of n points, representing transmitters/receivers. In the dual range assignment problem, we need to assign one of two ranges r_l and r_h to each point in P, such that the induced communication graph is either connected (in the symmetric model) or strongly connected (in the asymmetric model) and in addition some value is optimized. In this paper, we optimize the interference, assuming the asymmetric model. More precisely, the interference of a node in the induced directed communication graph is its indegree, and the interference of the graph is the maximum interference of a node of the graph. We prove that this version of the dual range assignment problem is NP-hard in the plane, and present an almost optimal solution on the line.

1 Introduction

Given a set P of n points in the plane, a dual range assignment is defined to be a function $\rho : P \to \{r_h, r_l\}$, where r_h and r_l represent high and low ranges, respectively. The communication graph in the asymmetric model is $G_\rho = (P, E)$, where $E = \{(p, q) \mid \rho(p) \geq |pq|\}$, where $|pq|$ is the Euclidean distance between p and q. Given a communication graph $G_\rho = (P, E)$, in the receiver-centric interference model, the interference of a point p, denoted by $RI(p)$, is defined as the number of points in $P \setminus \{p\}$ whose transmission range covers p, i.e., $RI(p) = |\{q \in P \setminus \{p\} : |pq| \leq \rho(q)\}|$. The receiver interference of G_ρ, denoted by $RI(G_\rho)$, is define as $max_{p_i \in P}\{RI(p_i)\}$. In the *Dual Min-Max Interference* problem, the goal is to find a range assignment ρ to the points of P such that the communication graph G_ρ induced by P and ρ is strongly connected, and $RI(G_\rho)$ is minimized.

1.1 Problem background

Limiting the interference between nodes in a sensor network is fundamental for the energy-efficiency of the network. Let P be a set of sensors in the plane. Each sensor $p \in P$ is assigned a transmission range $\rho(p)$. A sensor q can receive a signal from a sensor p if and only if q lies in the transmission area of p. There are two common ways to model the induced communication graph: symmetric and asymmetric. In the *symmetric* model the communication graph contains an edge between two points p and q if and only if both p and q lie in the transmission range of the other point; see Figure 1(b). For a valid assignment of transmission ranges, we require that the communication graph is connected. In the *asymmetric* model, the communication graph is directed, and there is an edge from p to q if and only if q lies in the transmission range of p; see Figure 1(a). In this case, we say that the assignment is valid if the communication graph is strongly connected.

In both the symmetric and the asymmetric case, the *receiver centric* interference of a point, denoted by $RI(p)$, is defined as the number of transmission ranges that it lies in; see Figure 2.

In the one-dimensional case, i.e., when the points are located on a line, Von Rickenbach et al. [10] showed that one can always construct a network with $O(\sqrt{n})$ interference, in the symmetric model. Moreover, they showed that there exists an instance that requires

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

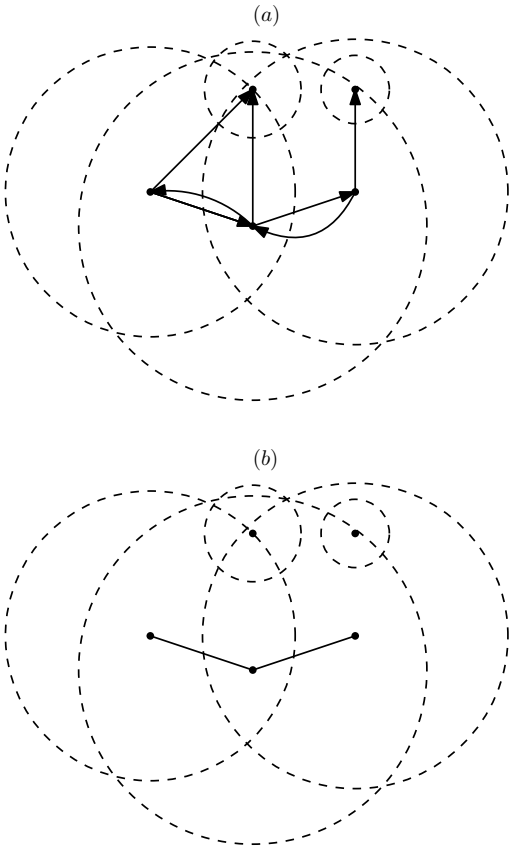


Figure 1 Illustration of asymmetric and symmetric models.

$\Omega(\sqrt{n})$ interference, they gave an $O(n^{1/4})$ approximation algorithm for this case. Tan et al. [9] proved that the optimal network has some interesting properties and, based on these properties, one can compute a network with minimum interference in sub-exponential time. Brise et al. [1] extend this result for the asymmetric model.

In the two-dimensional case, the problem has been shown to be NP-hard [1, 2]. Halldórsson and Tokuyama [5] showed how to construct a network with $O(\sqrt{n})$ interference for the symmetric model. For the asymmetric model, Fussen et al. [6] showed that one can always construct a network with $O(\log n)$ interference they showed that this algorithm is asymptotically optimal.

In this paper, the problem of minimizing the maximum receiver interference is considered where the transmission range of each vertex is limited to only two power levels i.e., high and low. Let r_h and r_l denote the transmission ranges of the high- and low-transmission powers, respectively. Since assigning more wireless nodes with the high power level results in a larger power consumption, the objective in the dual power assignment problem is equivalent to minimizing the number of wireless nodes that are assigned high-transmission range r_h and the induced graph to be strongly connected. The dual power assignment (DPA) problem was shown to be NP-hard[3, 8]. Rong et al [8] gave a 2-approximation algorithm, while Carmi and Katz in [3] gave a 9/5- approximation algorithm and a faster 11/6-approximation algorithm. Later, Chen et al. [4] proposed an $O(n^2)$ time algorithm with approximation ratio of 7/4.

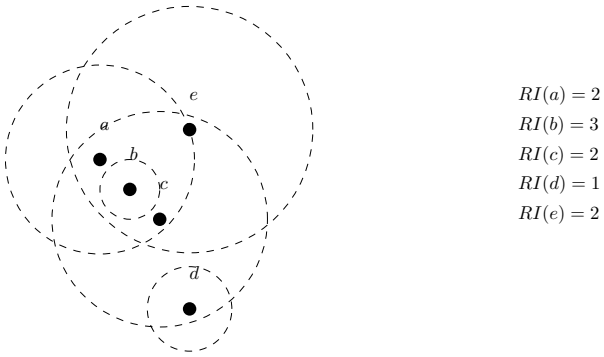


Figure 2 Illustration of receiver interference

Our results: We study the Dual Min-Max Interference Problem. We show that the problem is NP-complete for points in the plane. Then, when the points are located on a line, we give an almost optimal solution.

2 Dual Min-Max Interference Problem in 2D

In this section, we prove that The Asymmetrical Dual Min-Max problem in 2D is NPC. Our proof is based on [1] with a necessary modifications to the dual variant which make it much simple and elegant.

▶ **Theorem 2.1.** *The Dual Min-Max Interference Problem in 2D is NP-complete*

Proof. Given a range assignment ρ, it is easy to verify in polynomial time whether the graph induced by ρ is strongly connected and whether its maximum interference is bounded by a given value k. This implies that our problem is in NP.

To prove hardness of Min-Max Problem in $2D$, we show a polynomial time reduction from the problem of deciding whether a grid graph G of maximum degree 3 contains a Hamiltonian cycle, which is known to be NP-hard [7]. A grid graph is a graph whose vertex set is a subset of the integer grid $\mathbb{Z} \times \mathbb{Z}$, and two vertices are connected by an edge if and only if the distance between them is equal to 1. Note that we may assume that G is connected, otherwise there can be no Hamiltonian cycle.

Let $G = (V, E)$ be a grid graph with maximum degree 3, where $V = \{v_1, v_2, \ldots, v_n\}$. We construct in polynomial-time a set P of $8n$ points in the plane, and show that G contains a Hamiltonian cycle iff there exists a range assignment ρ such that the induced graph is strongly connected and $RI(G_\rho)$ is 5. Our reduction proceeds by replacing each vertex v of the given grid graph G by a *vertex gadget*; see Figure 3.

The vertex gadget consists of 8 points, and it has three parts: (a) the main point M with the same coordinates as v; (b) three connectors p_r, p_j, and p_w located on the grid in distance $1/4$ from M so that there is a connector for each edge in G that is incident to v (if v has degree two, the third connector can be placed in any of the two remaining directions), and (c) a set I, consisting of four points, the central point I_c lies at the remaining position in distance $1/4$ from v, and the other three points are located in distance ϵ from I_c; see Figure 3(c). For each v_i we define the gadget of v_i to be $P_i = \{M_i, p_{i_r}, p_{i_j}, p_{i_w}, I_{i_c}, I_{i_r}, I_{i_j}, I_{i_w}\}$. Finally, let $r_h = 1/2$, $r_l = 1/4$, and $k = 5$.

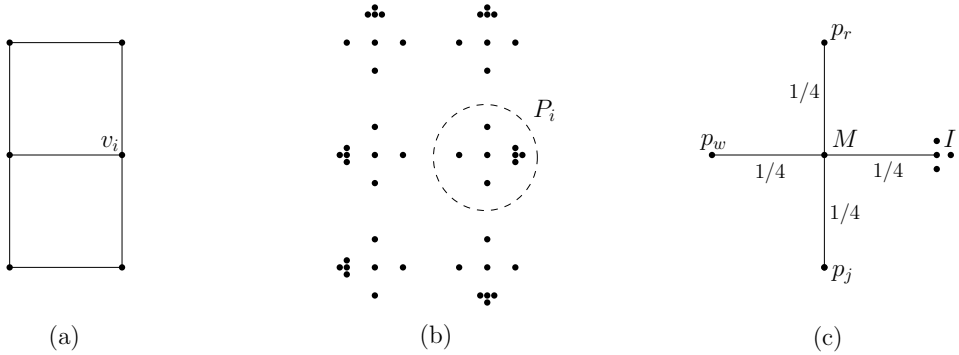


Figure 3 (a) A grid graph G, (b) the resulting set P, and (c) a gadget $P_i \subseteq P$ corresponds to the vertex v_i in G.

▶ **Lemma 2.2.** *Let ρ be a range assignment of P such that G_ρ is strongly connected and $RI(G_\rho)$ is 5. Then for each vertex gadget, there exists* exactly *one connector p that is assigned r_h.*

Proof. Since G_ρ is strongly connected, there exists at least one connector that is assigned r_h to get connectivity. We notice that in case we assign r_l for all points the interference of I_c for each gadget is 4. Since there exists at least one connector that is assigned r_h, we deduce that the interference of I_c is 5. Now, because we assumed that $RI(G_\rho)$ is 5, we get that *exactly* one connector p which assigned with r_h. ◀

We now prove the correctness of the reduction. Suppose that G contains a Hamiltonian cycle C. We compute a range assignment ρ to the points of P, such that G_ρ is strongly connected and $RI(G_\rho)$ is 5. Consider C as directed cycle, such that each vertex in C has in-degree 1 and out-degree 1. For each vertex v_i in G, we assign ranges to the points of P_i as follows. We assign r_l to the center M ,assign r_h to one of the connectors (according to the outgoing edge incident to v_i in C), and assign r_l to each of the other two connectors, and to all points in set I. Since C is a Hamiltonian cycle, the graph induced by ρ is strongly connected. Moreover, $RI(G_\rho)$ is 5.

Conversely, suppose that there exists a range assignment ρ to the points of P, such that G_ρ is strongly connected and $RI(G_\rho)$ is 5. By Lemma 2.2, exactly one connector p is assigned r_h and each of the other connectors p' is assigned r_l. We construct a Hamiltonian cycle C in G as follows. For every two sets P_i and P_j , we add the edge (v_i, v_j) to C if and only if the connector $p \in P_i$ that is assigned r_h covers a point in P_j . Thus, C is a subgraph of G, and, since G_ρ is strongly connected, C is connected. Moreover, since each set P_i has exactly one point which reaches a point not in P_i, the degree of each vertex in C is exactly 2. Therefore, C is a Hamiltonian cycle in G. ◀

3 Dual Min-Max Interference Problem in 1D

Let P be a set of n points located on a line and representing n transmitters-receivers. Let ρ_l denote a dual range assignment in which each point $p \in P$ is assigned $\rho(p) = r_l$. Let G be the graph induced by ρ_l. Let $M_1, M_2, \ldots, M_\ell$ be the strongly connected components of G sorted from left to right. For each component M_i, we define the sets $M_i^L = \{p \in M_i | \exists q \in M_{i-1}$ $|pq| \leq r_h\}$ and $M_i^R = \{p \in M_i | \exists q \in M_{i+1}$ $|pq| \leq r_h\}$.

We show that, if there exists a range assignment $\rho : P \to \{r_l, r_h\}$, such that the induced graph G_ρ is strongly connected and $RI(G_\rho) = k$, then one can compute a range $\rho' : P \to \{r_l, r_h\}$, such that the induced graph $G_{\rho'}$ is strongly connected and $RI(G_{\rho'}) \leq k+2$. Thus, we can perform a binary search on the possible values of k. For each value k, we use Algorithm 1 to decide whether there is a solution for this value. We return the range assignment obtained by the smallest k for which the algorithm does not return $FALSE$.

Algorithm 1 Decision Algorithm$(P, \{r_l, r_h\}, k)$

$\rho_l \leftarrow$ a dual range assignment in which each point $p \in P$ is assigned $\rho(p) = r_l$
Let $M_1, M_2, \ldots, M_\ell$ be the strongly connected components of G sorted from left to right
for $i \leftarrow 1$ to ℓ **do**
  if there exists a point $p \in M_i^L \cap M_i^R$ (if $i = 1$ then $p \in M_i^R$, and if $i = \ell$ then $p \in M_i^L$) such that, by assigning r_h to p, $RI(p') \leq k$, for each $p' \in M_i$ **then**
    assign r_h to the leftmost point p satisfying the condition
    update $RI(p')$ for each $p' \in P$
  else
    if there exist two points $p \in M_i^L$ and $q \in M_i^R$, such that, by assigning r_h to p and to q, $RI(p') \leq k$, for each $p' \in M_i$ **then**
      assign r_h to the leftmost point in M_i^L and to the leftmost point in M_i^R that satisfy the condition
      update $RI(p')$ for each $p' \in P$
    else
      return $FALSE$

Let ρ^* be an optimal range assignment of P and let G_{ρ^*} be the graph induced by ρ^*. Let ρ be the range assignment returned by Algorithm 1 and let G_ρ be the graph induced by ρ. In the following, we show that $RI(G_\rho) \leq RI(G_{\rho^*}) + 2$.

▶ **Observation 3.1.** *Let p be the rightmost point assigned $\rho(p) = r_h$ in M_i. Then, the rightmost point in M_i that is assigned r_h in ρ^* is to the left of p.*

▶ **Observation 3.2.** *The number of points in M_i that are assigned r_h in ρ^* is at least as the number of points in M_i that are assigned r_h in ρ.*

▶ **Theorem 3.3.** $RI(G_\rho) \leq RI(G_{\rho^*}) + 2$.

Proof. Let $p \in M_i$ and let $q \in M_j$, s.t. $j > i$, the most right point that assigned r_h and increase the receiver interference of p. The points which assigned r_h in the components between M_i and M_j in the optimal solution will also influence p, because if q reaches p then they will reach p too; see Figure 4. From Observation 3.2 we get that for each component

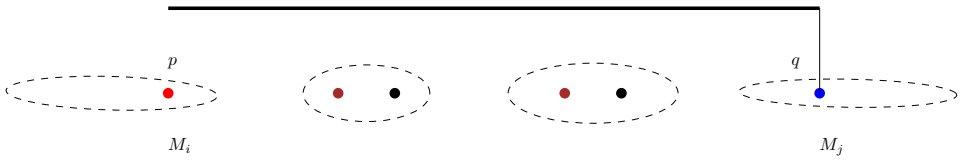


Figure 4 The red point represent p, the blue represent q. The brown points represents our algorithm r_h assignments, and the black represent the OPT r_h assignments.

between M_i and M_j the number of r_h which increase the interference of the point p is less

or equal to OPT. Now, let $w \in M_r$, s.t. $r < i$, be the most left point assigned with r_h and increase p receiver interference. The r_h assignments points in the components between M_p and M_r in the optimal solution will also influence p, because if w reaches p then they will reach p too; see Figure 5.

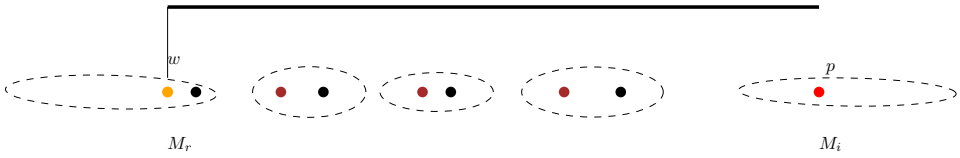


Figure 5 The red point represent p, the orange point represent r. The brown points represents our algorithm r_h assignments, and the black represent the OPT r_h assignments.

From Observation 3.2 we get that for each component between M_r and M_i the number of r_h which increase the interference of the point p is less or equal to OPT. From Observations 3.1 and 3.2 we get that we have less or equal number of r_h in component M_r that increase the interference of p. Therefore, the only components which can have a points s.t. increase p interference more them OPT is M_i and M_j. If the algorithm assign exactly one r_h in M_j or M_r then the claim that each of the components increase the interference by most one than OPT is obvious. Now assume that our algorithm assign two r_h in these components. From Observation 3.2 we know that the number r_h assignment by OPT will also be at least two. We notice that since our Algorithm 1 assign two r_h it means that do not exist a points $x \in M_i^L \cap M_i^R$ and $y \in M_j^L \cap M_j^R$. Therefore, in both algorithms we have one point which assigned r_h that belong to M_i^L or M_j^L and other that belong to M_i^R or M_j^R. We conclude that any two r_h assignments by our algorithm are not adjacent and at least one of the r_h assignment of the OPT algorithm was assigned to a point between the two points assigned by our algorithm. If the two r_h assignments in the algorithm increase p interference, then at least one out of the two r_h assignments by the OPT algorithm will also increase p interference. Therefore, for each M_j and M_r r_h assignments can increase the interference by at most one more then OPT. ◀

References

1 Yves Brise, Kevin Buchin, Dustin Eversmann, Michael Hoffmann, and Wolfgang Mulzer. Interference minimization in asymmetric sensor networks. In Jie Gao, Alon Efrat, Sándor P. Fekete, and Yanyong Zhang, editors, *Algorithms for Sensor Systems - 10th International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics, ALGOSENSORS 2014, Wroclaw*, volume 8847 of *Lecture Notes in Computer Science*, pages 136–151. Springer, 2014. doi:10.1007/978-3-662-46018-4_9.

2 Kevin Buchin. Minimizing the maximum interference is hard. *CoRR*, abs/0802.2134, 2008. URL: http://arxiv.org/abs/0802.2134, arXiv:0802.2134.

3 Paz Carmi and Matthew J. Katz. Power assignment in radio networks with two power levels. *Algorithmica*, 47(2):183–201, 2007. doi:10.1007/s00453-006-1230-1.

4 Jian-Jia Chen, Hsueh-I Lu, Tei-Wei Kuo, Chuan-Yue Yang, and Ai-Chun Pang. Dual power assignment for network connectivity in wireless sensor networks. In *Proceedings of the Global Telecommunications Conference, 2005. GLOBECOM '05, St. Louis, Missouri, USA, 28 November - 2 December 2005*, page 5. IEEE, 2005. doi:10.1109/GLOCOM.2005.1578450.

5 Magnús M. Halldórsson and Takeshi Tokuyama. Minimizing interference of a wireless ad-hoc network in a plane. *Theor. Comput. Sci.*, 402(1):29–42, 2008. doi:10.1016/j.tcs.2008.03.003.

6 Thomas Moscibroda and Roger Wattenhofer. Minimizing interference in ad hoc and sensor networks. In Suman Banerjee and Samrat Ganguly, editors, *Proceedings of the DIALM-POMC Joint Workshop on Foundations of Mobile Computing, Cologne, Germany, September 2, 2005*, pages 24–33. ACM, 2005. doi:10.1145/1080810.1080816.

7 Christos H. Papadimitriou and Umesh V. Vazirani. On two geometric problems related to the traveling salesman problem. *J. Algorithms*, 5(2):231–246, 1984. doi:10.1016/0196-6774(84)90029-4.

8 Yanxia Rong, Hongsik Choi, and Hyeong-Ah Choi. Dual power management for network connectivity in wireless sensor networks. In *18th International Parallel and Distributed Processing Symposium (IPDPS 2004), CD-ROM / Abstracts Proceedings, 26-30 April 2004, Santa Fe, New Mexico, USA*. IEEE Computer Society, 2004. doi:10.1109/IPDPS.2004.1303267.

9 Haisheng Tan, Tiancheng Lou, Yuexuan Wang, Qiang-Sheng Hua, and Francis C. M. Lau. Exact algorithms to minimize interference in wireless sensor networks. *Theor. Comput. Sci.*, 412(50):6913–6925, 2011. doi:10.1016/j.tcs.2011.08.041.

10 Pascal von Rickenbach, Roger Wattenhofer, and Aaron Zollinger. Algorithmic models of interference in wireless ad hoc and sensor networks. *IEEE/ACM Trans. Netw.*, 17(1):172–185, 2009. URL: http://doi.acm.org/10.1145/1514070.1514084, doi:10.1145/1514070.1514084.

Outerstring graphs of girth at least five are 3-colorable*

Sandip Das[1], Joydeep Mukherjee[2], and Uma kant Sahoo[1]

1 **Indian Statistical Institute, Kolkata.**
sandipdas@isical.ac.in, umakant.iitkgp@gmail.com
2 **Ramakrishna Mission Vivekananda Educational and Research Institute, Belur.**
joydeep.m1981@gmail.com

Abstract

An outerstring graph is the intersection graph of curves in a halfplane with one endpoint on the boundary of the halfplane. In an attempt to tackle a problem of Kostochka and Nešetřil (European J. of Comb. 19, 1998), we study coloring of outerstring graphs with girth $g \geq 5$, improving upon the latest bound. In the process, we generalize the results of Ageev (Discrete Math. 195, 1999), and Esperet and Ochem (Discrete Math. 309, 2009) on circle graphs.

1 Overview

The intersection graph of a family of sets $\mathcal{S}$ is the graph with vertex set $\mathcal{S}$ and edge set $\{XY \mid X, Y \in \mathcal{S}, X \neq Y, X \cap Y \neq \emptyset\}$. Here $\mathcal{S}$ is known as the *intersection representation* of G. A *string graph* is the intersection graph of a finite collection of *curves* or *strings*[1] in the plane. If one restricts that two curves in a string representation intersect at most once, then we call them *1-string* (denoted $\mathcal{S}_1$). We assume that whenever two strings intersect they *cross* each other. A well-known subclass of string graphs is the class of *outerstring graphs* (denoted $\mathcal{OS}$) where the strings are contained in a halfplane with exactly one point on the boundary of the halfplane and that point is an endpoint of the string.

Many interesting problems on the *vertex chromatic number* (denoted χ) of intersection graphs of geometric objects have been studied. One such problem is its dependence on *girth* [1, 2, 4, 8, 7, 9].

Our motivation is to focus on a problem posed by Kostochka and Nešetřil [8] on coloring 1-string graphs of girth five. In this regard, we show that *outerstring* graphs of girth g at least five and minimum degree at least two have $(g-4)$ vertices of degree two that induce a path. As we shall see, a corollary of this result is a first step towards our main goal. Furthermore, this seems to be a natural milestone as it generalizes similar results on *circle graphs*, by Ageev [1], and by Esperet and Ochem [4].

Borrowing notation from Kostochka and Nešetřil [8], given a class of intersection graphs $\mathcal{G}$, let

$$\chi(\mathcal{G}, k) := \max_{G \in \mathcal{G}} \{\chi(G) \mid \text{girth}(G) \geq k\}.$$

In 1970s[2], Erdős (see [6, Problem 1.9]) posed whether $\chi(\mathcal{I}, 4) < \infty$, where $\mathcal{I}$ is the class of intersection graphs of line segments in the plane. Further, Kratochvíl and Nešetřil (see [7]) posed whether $\chi(\mathcal{S}_1, 4) < \infty$. Recently, in a breakthrough paper, Pawlik et al. [9] proved

* Partially supported by the IFCAM project MA/IFCAM/18/39.

[1] A curve or string is a homeomorphic image of the interval $[0, 1]$ in the plane.

[2] An approximate date was confirmed in a personal communication with András Gyárfás and Janós Pach to the authors of [9] (Pawlick et al.). See footnote 2 in Pawlick et al. [9].

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

that $\chi(\mathcal{I}, 4)$ can be arbitrarily large, hence settling the questions of Erdős (see [6]), and of Kratochvíl and Nešetřil (see [7]).

Earlier, motivated by the above problems, Kostochka and Nešetřil [8] studied 1-string graphs with girth at least five. In particular they proved that $\chi(\mathcal{S}_1, 5) \leq 6$. They also posed if $\chi(\mathcal{S}_1, 5) > 3$.

As a step in improving the bound of $\chi(\mathcal{S}_1, 5)$, we study $\chi(\mathcal{OS}, 5)$. Our main result implies that[3] $\chi(\mathcal{OS}, 5) = 3$, which is an immediate corollary of the following.

▶ **Theorem 1.1.** *Every outerstring graph with girth at least five is 2-degenerate.*

Theorem 1.1 is crucial to our approach to improve the bounds of $\chi(\mathcal{S}_1, 5)$. Given such a 1-string representation (with girth $g \geq 5$), we can treat its *outer envelope* as the stab line. As we shall see, the target degree two vertices in the outerstring graph induced by strings intersecting this *stab line* are in some sense *closest* to the stab line. This would result in finding a degree three vertex in the 1-string graph (because of the girth restriction). We shall address this in a future work.

Thus every outerstring graph, and its subgraphs, with girth at least five has a vertex of degree at most two, and hence is 3-colorable. This improves upon the previous best upper-bound of five due to Gavenčiak et al. [5]. Furthermore, all odd cycles are outerstring graphs.

▶ **Corollary 1.2.** $\chi(\mathcal{OS}, 5) = 3$.

Recently, Rok and Walczak [10] showed that outerstring graphs are chi-bounded. This implies that their chromatic number is upper-bounded by their clique numbers. In particular, the chromatic number of triangle-free outerstring graphs is upper-bounded by a constant. Corollary 1.2 shows that if we further forbid the presence of cycles of length four in this graph class, then the chromatic number of such graphs is upper-bounded by three.

The main result of this article, an extension of Theorem 1.1, seems to be a natural milestone in our approach to improve the upper bound of $\chi(\mathcal{S}_1, 5)$ as it generalizes similar results on circle graphs, first by Ageev [1], and then by Esperet and Ochem [4].

▶ **Theorem 1.3.** *(Ageev [1]) Every circle graph with girth at least five is 2-degenerate.*

▶ **Theorem 1.4.** *(Esperet and Ochem [4]) Every circle graph with girth g at least five and minimum degree δ at least two contains a chain of $g - 4$ vertices of degree two.*

In this article, we prove the following extension of Theorem 1.4 to outerstring graphs. It is not difficult to see that outerstring graphs strictly contain circle graphs. There are $2^{\Theta(n \lg n)}$ (labeled) circle graphs whereas there are $2^{\Theta(n^2)}$ (labeled) outerstring graphs on n vertices[4]. See Figure 1 for a separating example between circle graphs and outerstring graphs with girth five and minimum degree two. The proof is omitted for lack of space.

▶ **Theorem 1.5.** *Every outerstring graph with girth $g \geq 5$ and minimum degree δ at least two contains a chain of $g - 4$ vertices of degree two.*

Note that Theorem 1.1 directly follows from Theorem 1.5. Hence we shall only prove Theorem 1.5. Also, it suffices to consider only connected outerstring graphs.

---

[3] In this context, although it suffices to prove that $\chi(\mathcal{OS}_1, 5) = 3$, we prove a more general result.
[4] see the corresponding `graphclasses` pages at `www.graphclasses.org`.

(a) **(b)**

Figure 1 (a) Separating Example G_5 and (b) Outerstring representation of G_5.

2 Definitions and Notations

Basic Notions. For an outerstring graph OS, we denote its outerstring representation as $\mathbb{OS}$. Due to a result of Biedl, Biniaz and Derka [3], we can choose $\mathbb{OS}$ such that it has finite number of intersection points. We assume the stab line in $\mathbb{OS}$ to be the Y-axis and the strings lie in the halfplane $x \leq 0$. For each string s_i in $\mathbb{OS}$, its endpoint on the stab line is its *fixed end* and the other endpoint is its *free end*. We can safely assume that all the free ends of the strings are intersection points (see Figure 2a). We relax the *crossing* assumption at these points. Since the stab line is the Y-axis, we can order all the fixed ends of strings; so the terms *above*, *below*, *topmost* and *bottommost* are well defined.

As any two strings, say s_i and s_j, might intersect multiple times in $\mathbb{OS}$, we say all such intersection points are of *type* (s_i, s_j).

A region $\mathcal{A} \subset \mathbb{R}^2$ is *arc-connected* if for any two points in $\mathcal{A}$ there is a curve lying completely in $\mathcal{A}$ that connects them. A *face* in $\mathbb{OS}$ is a maximal arc-connected component of $\mathbb{R}^2 \setminus \bigcup s_i$. Here the ambient space is $\mathbb{R}^2$ and not the halfplane $x \leq 0$. Since two strings can intersect multiple times, there are faces in $\mathbb{OS}$ which have just two strings in its boundary. We call such faces as 2-faces (see Figure 2b). An *arrangement* induced by a set of curves in $\mathbb{R}^2$ is the embedding of these curves in $\mathbb{R}^2$.

An *n-chain* in OS is a path of length $n+1$ in OS whose n internal vertices have degree 2 in OS. We shall also call the corresponding representation of this path in $\mathbb{OS}$ as a *n-chain*.

For a closed region $\mathcal{R}$, its inside region (after removing its boundary) is denoted as $Int(\mathcal{R})$.

Left Envelope. Consider an outerstring representation $\mathbb{OS}$ of an outerstring graph OS. Let t_0 (respectively, b_0) be the topmost fixed end (respectively, bottommost fixed end) of $\mathbb{OS}$. Let t_0b_0 be the line segment (on the stab line) from t_0 to b_0. Consider the arrangement induced by the strings in $\mathbb{OS}$ and t_0b_0 in $\mathbb{R}^2$, and consider the unbounded face of this arrangement. The *left envelope* of $\mathbb{OS}$ is the part of this boundary after removing t_0b_0 except its end points (see Figure 2c). We say a string s *belongs* to the left envelope $\mathcal{E}$ if $s \cap \mathcal{E} \neq \emptyset$.

Zone. In an outerstring representation, given two intersecting strings s_t and s_b, let the fixed end t of s_t be above the fixed end b of s_b. The *zone* of s_t and s_b in $\mathbb{OS}$ is the bounded face in $\mathbb{R}^2 \setminus \{s_t, s_b, tb\}$ that contains tb in its boundary (see Figure 2d). The intersecting strings s_t and s_b are called as the *defining strings* of this zone: with s_t as its *top defining string* and s_b as its *bottom defining string*. So every pair of intersecting strings correspond to a zone.

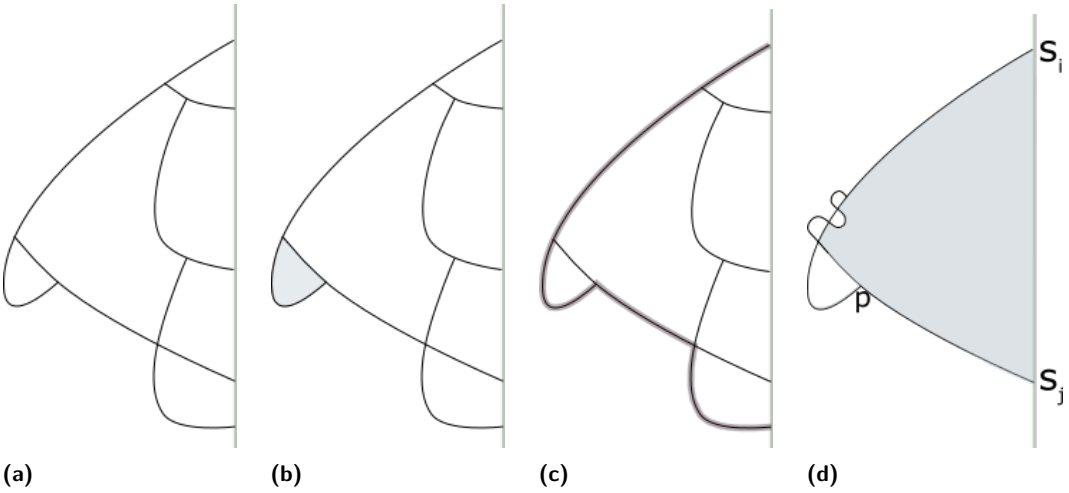


Figure 2 (a) An Outerstring representation, (b) a 2-face, (c) left envelope, and (d) a zone.

A zone $\mathcal{Z}$ of s_t and s_b is *filled* if there exists a string s in $\mathbb{OS}$ whose fixed end is in $\mathcal{Z}$ and s does not intersect with s_t and s_b. So $s \cap \mathcal{Z} = s$ and $s \cap s_t = s \cap s_b = \emptyset$: we say $\mathcal{Z}$ *supports* s.

k-Face. Consider an outerstring graph OS with girth $g \geq 5$. A *k-face* in $\mathbb{OS}$ is a face that contributes to a cycle in OS (see Figure 3a). The k corresponds to the size of the cycle. The strings corresponding to these k vertices are called the bounding strings of k-face. Note that not all faces in $\mathbb{OS}$ contribute to a cycle in OS, e.g. the 2-faces. We have an algorithm to check whether a face in $\mathbb{OS}$ is a k-face or not. We have the following surprising observation. The algorithm and the proof of the observation are omitted for lack of space.

▶ Observation 2.1. The vertices corresponding to the bounding strings of a k-face $\mathcal{F}$ in $\mathbb{OS}$ of an outerstring graph OS (with girth $g \geq 5$) form an induced cycle in OS.

Extended Face. For an induced cycle in OS with girth five, its induced representation in $\mathbb{OS}$ might not be a face but a collection of faces. Given an induced cycle C on k (≥ 5) vertices in OS, there is at least one closed curve in $\mathbb{OS}$ composed of the k strings corresponding to the vertices of C. Let $\mathcal{C}$ denote such a closed curve satisfying the following two conditions.

1. $\mathcal{C}$ is a minimal closed curve in the sense that there is no part of $\mathcal{C}$ that is a closed curve composed of the k strings. (We show that $\mathcal{C}$ is a Jordan curve. The proof is omitted for lack of space.)
2. Among all Jordan curves satisfying condition (1), choose $\mathcal{C}$ such that its inside region has the minimum area.

We say the closed inside region of $\mathcal{C}$ is the *extended face* of C (see Figure 3b and 3c for non-examples: see Figure 3d for an example).

3 Outline of Proof of Theorem 1.5

The proof of Theorem 1.5 consists of three parts. The proof is omitted for lack of space. Here we give a detailed outline. The main proof is by strong induction and is done in Section 3.3. To this end, we need to study the outerstring representations of cycles (for the base step

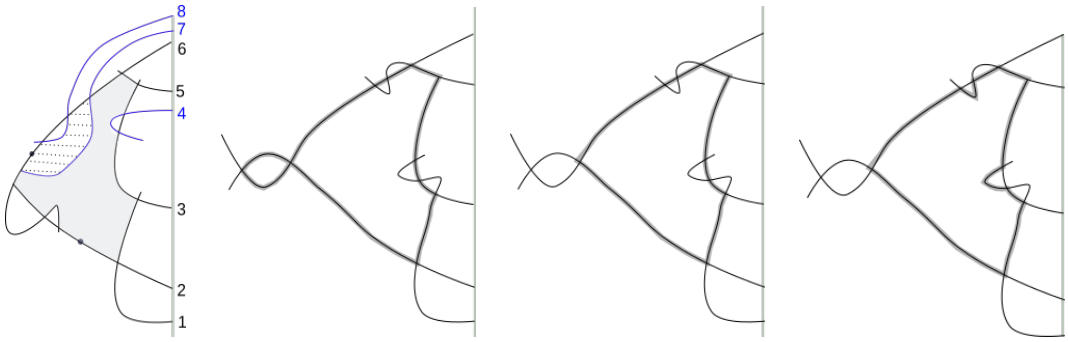


Figure 3 (a) 5-face (shaded) (b,c) non-examples of extended face (violates condition 1 & 2, and 2 respectively), and (d) an example of an extended face.

in induction) in Section 3.1, and some basic properties of outerstring representations in Section 3.2.

3.1 Representation of Cycles

Consider an outerstring representation $\mathbb{C}$ of a cycle C on $n \geq 5$ vertices. We first prove that if $\mathbb{C}$ has a filled zone $\mathcal{Z}$, then every string s in $\mathbb{C}$, other than the defining strings of $\mathcal{Z}$, has an intersection point in $Int(\mathcal{Z})$. Then we prove the following.

▶ Claim 3.1. There is at least one filled zone in $\mathbb{C}$. Every filled zone in $\mathbb{C}$ contains a $(n-4)$-chain.

We call such strings that are *supported* by the filled zone as *intermediate strings*. These intermediate strings induce a $(n-4)$-chain. While proving Claim 3.1, we found that $\mathbb{C}$ can have either one or three filled zones (see Figures 4a, 4b, 4c) depending on the number of types of intersection points in its left envelope.

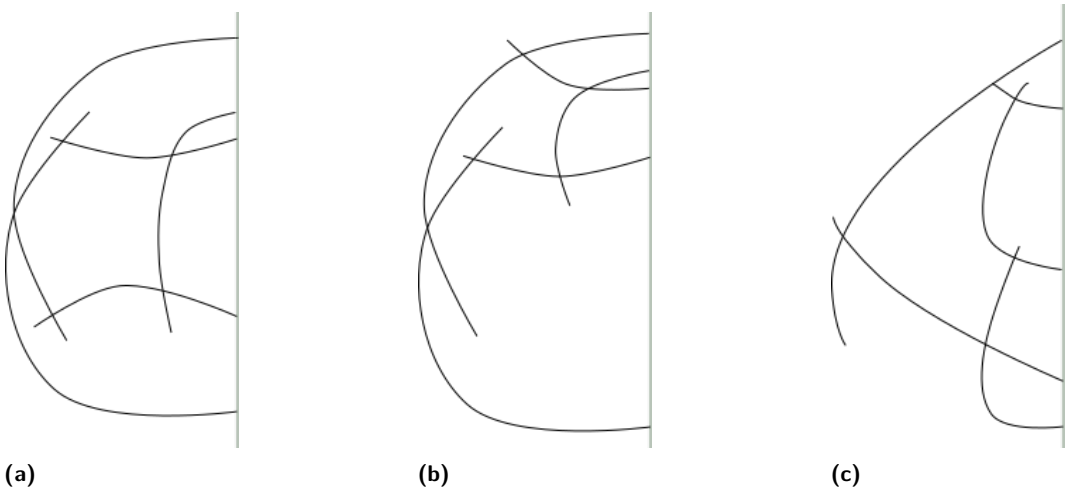


Figure 4 Three possibilities of $\mathbb{C}$ with one, two and three types of intersection points in the left envelope.

3.2 Some Basic Properties of Outerstring Representations

Consider an outerstring graph OS with girth $g \geq 5$ and minimum degree $\delta \geq 2$. Let $\mathcal{Z}$ be a filled zone in $\mathbb{OS}$. Such a filled zone always exists (see Claim 3.1). We prove the following.

▶ Subclaim 3.2. For every filled zone $\mathcal{Z}$ in $\mathbb{OS}$, there exists a k-face in $\mathcal{Z}$.

A key argument used in the proof of Subclaim 3.2, as well as in the main proof in the next subsection, is called the *branching argument*. If a filled zone is not allowed to have a k-face in it and $\delta \geq 2$, then consider the forest/tree induced by the strings supported by the zone. The leaves (except one) of this tree has degree one in OS, else a k-face is formed in the filled zone. This contradicts our minimum degree restriction. Hence Subclaim 3.2 follows.

In the rest of this section, we prove the following stronger version of Theorem 1.5.

▶ Claim 3.3. Given an outerstring graph OS with girth $g \geq 5$ and $\delta \geq 2$, any filled zone $\mathcal{Z}$ in $\mathbb{OS}$ completely contains a $(g-4)$-chain, that is, the strings in this chain are supported by $\mathcal{Z}$.

3.3 Outline of Proof of Claim 3.3

We proceed by induction on the number of vertices in OS. The base case is easy to verify, as it is a cycle on g vertices (see Claim 3.1). Now assume the induction hypothesis: in any outerstring graph on less than l vertices with girth $g \geq 5$ and $\delta \geq 2$, any filled zone in its outerstring representation completely contains a $(g-4)$-chain. Let OS be an outerstring graph on l vertices with girth $g \geq 5$ and $\delta \geq 2$. Consider a filled zone $\mathcal{Z}$ in $\mathbb{OS}$. Our aim is to find a $(g-4)$-chain in the filled zone $\mathcal{Z}$ in $\mathbb{OS}$. The rest of the proof relies heavily on the following subclaim, which repeatedly requires the Subclaim 3.2 and the branching argument.

▶ Subclaim 3.4. We can safely assume that every string in $\mathbb{OS}$, except the defining strings of $\mathcal{Z}$, has an intersection point in $Int(\mathcal{Z})$.

We can also assume that OS is not isomorphic to the cycle on l vertices. Indeed, we have proved Claim 3.3 for cycles (see Claim 3.1). Hence there are at least two (induced) cycles in OS. Next, we prove the following.

▶ Subclaim 3.5. $\mathbb{OS}$ has at least two filled zones.

Next, using Subclaim 3.5, we show that two filled zones are restricted to attain one of two possible configurations. In each of these possible configurations, we exhibit a filled zone and a string that does not have an intersection point in the interior of the filled zone, thereby contradicting Subclaim 3.4. This completes the proof of Claim 3.3. ◀

4 Acknowledgements

The authors thank the reviewers for enhancing both the content and the presentation of this paper. In particular, we thank them for the implications of the paper by Rok and Walczak [10], and for the details on the sizes of labeled circle graphs and outerstring graphs on n vertices. The authors are partially supported by IFCAM project Applications of graph homomorphisms (MA/IFCAM/18/39).

References

1 A. A. Ageev. Every circle graph of girth at least 5 is 3-colourable. *Discrete Mathematics*, 195(1):229–233, 1999. URL: http://www.sciencedirect.com/science/article/pii/S0012365X98001927, doi:10.1016/S0012-365X(98)00192-7.

2 E. Asplund and B. Grünbaum. On a Coloring Problem. *Mathematica Scandinavica*, 8:181–188, 1960. URL: https://www.mscand.dk/article/view/10607, doi:10.7146/math.scand.a-10607.

3 T. Biedl, A. Biniaz, and M. Derka. On the Size of Outer-String Representations. In D. Eppstein, editor, *16th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2018)*, volume 101 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 10:1–10:14, Dagstuhl, Germany, 2018. URL: http://drops.dagstuhl.de/opus/volltexte/2018/8836, doi:10.4230/LIPIcs.SWAT.2018.10.

4 L. Esperet and P. Ochem. On circle graphs with girth at least five. *Discrete Mathematics*, 309(8):2217–2222, 2009. URL: http://www.sciencedirect.com/science/article/pii/S0012365X08003099, doi:10.1016/j.disc.2008.04.054.

5 T. Gavenčiak, P. Gordinowicz, V. Jelínek, P Klavík, and J. Kratochvíl. Cops and Robbers on intersection graphs. *European Journal of Combinatorics*, 72:45–69, 2018. URL: http://www.sciencedirect.com/science/article/pii/S0195669818300817, doi:10.1016/j.ejc.2018.04.009.

6 A. Gyárfás. Problems from the world surrounding perfect graphs. *Applicationes Mathematicae*, 19(3-4):413–441, 1987. URL: https://eudml.org/doc/265405.

7 A. V. Kostochka and J. Nešetřil. Chromatic number of geometric intersection graphs. In *1995 Prague Midsummer Combinatorial Workshop*, pages 43–45, 1995.

8 A. V. Kostochka and J. Nešetřil. Coloring Relatives of Intervals on the Plane, I: Chromatic Number Versus Girth. *European Journal of Combinatorics*, 19(1):103–110, 1998. URL: http://www.sciencedirect.com/science/article/pii/S0195669897901517, doi:10.1006/eujc.1997.0151.

9 A. Pawlik, J. Kozik, T. Krawczyk, M. Lasoń, P. Micek, W. T. Trotter, and B. Walczak. Triangle-free intersection graphs of line segments with large chromatic number. *Journal of Combinatorial Theory, Series B*, 105:6–10, 2014. URL: http://www.sciencedirect.com/science/article/pii/S009589561300083X, doi:10.1016/j.jctb.2013.11.001.

10 A. Rok and B. Walczak. Outerstring Graphs are χ-Bounded. *SIAM Journal on Discrete Mathematics*, 33(4):2181–2199, 2019. URL: https://epubs.siam.org/doi/abs/10.1137/17M1157374, doi:10.1137/17M1157374.

Disjoint Box Covering in a Rectilinear Polygon*

Sujoy Bhore[1], Guangping Li[2], Martin Nöllenburg[2], and Jules Wulms[2]

1 **Université libre de Bruxelles, Belgium**
sujoy.bhore@gmail.com
2 **Algorithms and Complexity Group, TU Wien, Vienna, Austria**
{guangping, noellenburg, jwulms}@ac.tuwien.ac.at

Abstract

We study a variant of the geometric covering problem, namely, *Disjoint Box Covering in a Rectilinear Polygon* (DBCR). Given a rectilinear polygon R with m edges and a finite set P of n points inside R, in the DBCR problem, the objective is to find a set $\mathcal{S}$ of pairwise disjoint axis-aligned rectangles, such that $\mathcal{S}$ covers P, each rectangle is fully contained inside the polygon R, and $\mathcal{S}$ has minimum cardinality. We show that this problem is NP-hard for points in general position. Moreover, we design an $O(n \log n + m \log m)$-time exact algorithm for the DBCR problem if the input rectilinear polygon is a histogram.

1 Introduction

In the geometric set cover problem, the input is a range space $\Sigma = (X, \mathcal{R})$, where X is a universe of points in $\mathbb{R}^d$ and $\mathcal{R}$ is a family of subsets of X called ranges. The subsets in $\mathcal{R}$ are defined by the intersection of X and geometric shapes such as axis-parallel rectangles, disks, etc. The objective is to select a minimum-size subset $\mathcal{C} \subseteq \mathcal{R}$ of ranges such that every point in the universe X is covered by some range in $\mathcal{C}$. The geometric set cover problem is known to be NP-complete even for simple geometric range spaces, e.g., when $\mathcal{R}$ is a set of unit disks or unit squares in $\mathbb{R}^2$ [16]. However, the problem is far more tractable in the geometric domain than the classical set cover problem. For instance, while it is known that the set cover problem is inapproximable within factor $O(\log n)$, there exist several polynomial time approximation schemes for the geometric variant of the problem that used underlying geometric structures to achieve better algorithmic results; see [1, 3, 12, 13].

We study a special variant of the geometric set cover problem, namely, the *Disjoint Box Covering in a Rectilinear Polygon* (DBCR) problem. Given a rectilinear polygon R, possibly with holes, and a finite set P of n points inside R, in the DBCR problem, the objective is to find a set $\mathcal{S}$ of pairwise disjoint axis-aligned rectangles, such that $\mathcal{S}$ covers P and each rectangle is fully contained inside R. The DBCR problem is particularly motivated by geographically-informed text-based visualizations, such as spatial word or tag clouds. Imagine that we are given a set of locations, the point set P, inside a geographic region, the polygon R, with several possible text labels per location, which may refer to different categories of P (colored points). It is important for spatial word clouds to not only capture the frequency or weight but additionally the spatial relevance of the words (see [9]). If we model the words as axis-parallel rectangles, then the objective is to cover P with disjoint rectangles such that each rectangle contains only points of the same color and stays inside R. In this work, we address this problem under the simplifying assumption that the points are inside a rectilinear polygon and are all of same color. Without any color restrictions, one

* Research supported by the Austrian Science Fund (FWF) under grant P31119.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

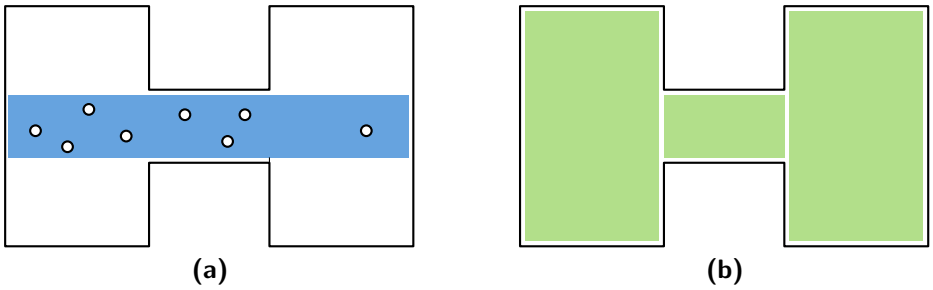


Figure 1 Comparing a solution to DBCR in **(a)** by a partitioning of the polygon in **(b)**.

way to solve the problem is to compute a minimum rectangular partitioning of the input polygon, which is solvable in polynomial time [18]. However, in a U-shaped polygon the solution of the partitioning problem could already differ from the solution of the DBCR problem (see Fig. 1). In fact, the DBCR problem is NP-hard if R is an arbitrary rectilinear polygon (Section 2). However, if R is a histogram, i.e., a rectilinear polygon consisting of a base line and a monotone path, we can give a polynomial-time algorithm (Section 3).

Related work. Many variants of the geometric covering problem have been investigated over the years, e.g., (p, k)-box covering [2]; class cover problems [8], red-blue cover [5, 11, 10]. Moreover, geometric covering problems have been widely studied in various algorithmic paradigms such as approximation algorithms (see [1, 13, 6, 14]) and parameterized algorithms (see [4, 7]). However, most of these works do not consider the case where the output objects must be disjoint. This makes the DBCR problem particularly unique, and none of these algorithms can be directly applied in our context.

2 NP-completeness of DBCR

In this section, we show firstly that the DBCR problem is NP-complete. We do a reduction similar to the one by Chan and Hu for red-blue set cover [11], using the following Lemmata.

▶ **Lemma 2.1.** *([19]) Every planar graph $G = (V, E)$ of maximum degree at most 4 has an orthogonal grid drawing on an $O(|V|) \times O(|V|)$ grid.*

▶ **Lemma 2.2.** (*Folklore*) *Given a graph G and an edge e in G, define a new graph G' obtained from G by subdividing e through the addition of two new vertices. Then the size of a minimum vertex cover of G' is exactly the size of a minimum vertex cover of G plus 1.*

▶ **Theorem 2.3.** *Disjoint Box Covering in a Rectilinear Polygon is NP-hard.*

Proof. We reduce from the vertex cover problem on 3-regular planar graphs, which is known to be NP-hard by Garey and Johnson [17]. Given a 3-regular planar graph G with n vertices, we create an orthogonal grid drawing δ_G by Lemma 2.1; see Figure 2a as an illustration. In the following, we construct a new graph G' by adding several dummy vertices in δ_G.

Firstly, we add a dummy vertex on each bend of δ_G, to create a straight-line drawing. We call a vertex v in δ_G a *corner* vertex, if there exists a pair of perpendicular line segments in δ_G that meet at v. Then, for each edge e connecting two corner vertices, we add a dummy vertex in the middle of e. This results in the new graph G', where each edge is incident to at most one corner vertex. To apply Lemma 2.2, for each original edge e in G, we check whether there is an even number of dummy vertices on e, including dummy corner vertices.

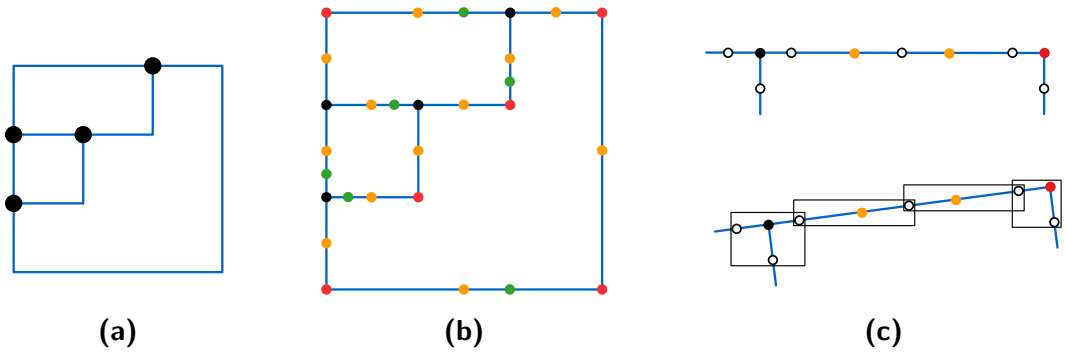


Figure 2 The reduction on graph $G = K_4$. **(a)** An orthogonal grid drawing δ_G of G. **(b)** The dummy vertices are added in δ_G to the bends (red) and on the edges, both to prevent multiple corner vertices incident to an edge (yellow), and for parity (green). **(c)** Cover rectangles, for black, red, yellow and green vertices, cover all (white) points in P and are drawn on the shifted grid.

If not, we then add a dummy vertex anywhere on the edge, to ensure the vertex cover of G' changes predictably (see Figure 2b).

To construct point set P in the DBCR instance from G', we replace each edge in G' by a point. If an edge e is incident to a corner vertex v, we put a point at $\frac{1}{3}$ of grid length from v on e. Otherwise, we create a point in the middle of the edge. Now we shift the points such that there are no two points on the same horizontal or vertical line. To achieve this, we map the drawing $\delta_{G'}$ to an orthogonal grid rotated by α. We look at each corner vertex c in $\delta_{G'}$ and consider each pair of orthogonal edges incident to c with lengths x_c and y_c. We choose $\alpha < \min\left\{\arctan(\min_c \frac{x_c}{y_c})/2, \arctan(\min_c \frac{y_c}{x_c})/2\right\}$. For each vertex v in G', we draw a rectangle B_v, such that B_v is the minimum bounding box covering v and the points on edges of v (see Figure 2c). By our choice of α these rectangles overlap only at points in P.

We say that B_v is the *cover rectangle* of the vertex v. For each corner vertex, we add a margin of length $\frac{1}{2}\epsilon$ to its cover rectangle, where ϵ is an infinitesimally small distance. This ensures that the covered points are not on its boundaries and the rectangles grow by ϵ both horizontally and vertically. Intuitively, there are two types of cover rectangles, small rectangles for corner vertices and long rectangular bars otherwise. We can then make the following observations.

▶ **Observation 2.4.** *Two cover rectangles intersect if and only if its corresponding vertices are adjacent; For each edge of G', only the rectangles of its two incident vertices intersect with it and the intersection contains the point on the edge.*

The union of cover rectangles builds the rectilinear polygon R in our DBCR instance.

▶ **Observation 2.5.** *Given an arbitrary rectangle S in R that covers a subset P' of point set P, the minimum bounding box of P' is inside a cover rectangle.*

After shifting, the points (and the corresponding cover rectangles) on a horizontal/vertical grid line are in a monotonic order. For two points that are not covered by the same rectangle B_v of any vertex v, there is no rectangle inside P that covers these points at the same time.

The correctness of the reduction is now straightforward. A vertex cover in G' corresponds to a set $\mathcal{S}$ of cover rectangles in R that covers all the points P. By Observation 2.4, if two cover rectangles intersect each other, their corresponding vertices are adjacent. Due to our construction, no two corner vertices are adjacent. Therefore, each intersection involves at

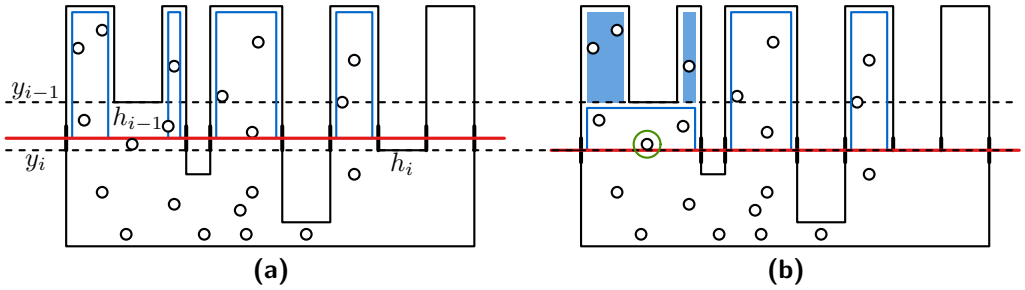


Figure 3 The red sweep line stores the vertical segments of the histogram (in bold) and the rectangles in the solution (in blue) at a certain y-coordinate. **(a)** The status is shown between events at y_{i-1} and y_i. **(b)** The point circled in green is found by querying Q_P at the event of h_i.

least one non-corner vertex. Consider a rectangles $S \in \mathcal{S}$ whose corresponding vertex is not a corner vertex. If the rectangle S intersects other rectangles in $\mathcal{S}$, we could eliminate the intersection by shrinking long rectangle S, while still covering all the points. After eliminating all the intersections in $\mathcal{S}$, it is a solution to the DBCR problem for point set P in R. In the other direction, we are given a set $\mathcal{S}$ of disjoint rectangles inside R that cover all the points in P. By Observation 2.5, each rectangle in $\mathcal{S}$ is completely inside a cover rectangle in the polygon R. We select all the cover rectangles that contains rectangles in $\mathcal{S}$. These rectangles cover all the points and their corresponding vertices in G' cover all the edges in G', thus they would build a vertex cover in G'. ◀

Containment in NP is straightforward for the DBCR problem, since we have to check only whether each point in P is covered by a rectangle in $\mathcal{S}$, and each rectangle in $\mathcal{S}$ is contained in R. These tasks can be done efficiently using point location data structures.

3 Covering points inside histogram

We solve the DBCR problem inside a *histogram*, which is defined as in [15] (see Figure 3).

Histogram: Define a (vertical) histogram with m edges as a rectilinear polygon with one horizontal edge (the base) equal in length to the sum of all other horizontal edges.

The DBCR problem in a histogram can be solved in polynomial time using a greedy algorithm $\mathcal{A}$. We assume the base of the input histogram H is located at $y = 0$ and extends only upwards, ending in $\frac{m}{2} - 1$ horizontal edges.

Algorithm $\mathcal{A}$ first sorts the $\frac{m}{2}$ horizontal pieces based on their y-coordinates in descending order, resulting in queue Q_H. Furthermore, the points in P are sorted by their y-coordinates, resulting in a queue Q_P. Using a horizontal sweep line l, an optimal solution $\mathcal{S}$ is constructed starting from the top of the histogram H. A *status* s stored with the sweep line maintains the x-coordinates of vertical rectangle edges in $\mathcal{S}$, as well as the x-coordinates of the vertical segments of H, at a certain y-coordinate (see Figure 3a).

As the sweep line l moves downwards, an event occurs at every horizontal segment h of H. At an event the status must be updated, and we query Q_H to find the next horizontal segment h_i of H. By querying Q_P, we can quickly find all points between h_i and the previous horizontal segment h_{i-1}, at y-coordinates y_i and y_{i-1}, respectively. If there are no points, then the solution $\mathcal{S}$ and status s require no change. However, if there are points, we check whether (at least) one of the points is not covered by the rectangles stored in s. If this is

the case for some point p, then we add a new rectangle to the solution $\mathcal{S}$, starting at y_{i-1} and spanning the whole width between the vertical segments of H neighboring p. These vertical segments can be found by querying s using the x-coordinate of p, and walking in both directions through s until a vertical segment is encountered. To prevent rectangles in $\mathcal{S}$ from overlapping, we settle and remove any rectangles from s that would otherwise overlap the newly added rectangle. The settled rectangles will vertically extend down to y_{i-1} in the final solution $\mathcal{S}$ (see Figure 3b).

Finally, depending on what type of segment h_i is, the status s must be updated accordingly. The segment h_i can have one of the following two types:

- Segment h_i is the top of a vertical spike of H. In this case, s is updated to include the x-coordinates of the vertical segments, pointing downwards from h_i.
- Segment h_i connects two vertical spikes of H. Status s must then be updated to no longer include the vertical segments that point downwards to h_i.

A complete solution found by algorithm $\mathcal{A}$ can be found in Figure 4.

▶ **Lemma 3.1.** *Algorithm $\mathcal{A}$ finds an optimal solution for the DBCR problem, given a set P of points inside a histogram H.*

Proof. We make a case distinction on whether an optimal solution OPT that has at least two rectangles r_1, r_2 horizontally next to each other without vertical histogram segments between them. First assume there are no such rectangles, and hence can extend each rectangle left- and rightwards until the polygon is hit, without intersecting any other rectangle. Next, we start from the base of the polygon and extend rectangles upwards until a horizontal edge of H is reached. Any rectangles intersected in the process shrink by moving their bottom edge upwards, leaving no points uncovered, since they will be covered by the rectangles that are extending upwards. The resulting solution is equivalent to a greedy solution of algorithm $\mathcal{A}$.

Now assume we have an optimal solution OPT that has at least two such rectangles r_1, r_2. Note that one of these rectangles may extend further upwards than the other. Assume w.l.o.g. that r_1 extends further upwards if this is the case.

Consider the greedy solution OPT', which instead contains rectangles r'_1, r'_2, such that r'_1 stops at the top of r_2 and r'_2 covers the whole space occupied by r_1, r_2 horizontally. Rectangle r'_2 can also extend as far downwards as the lowest boundary of r_1, r_2. This cannot lead to an intersection with the polygon boundary, because the histogram ends in a base that stretches the whole width horizontally. Furthermore, any intersected rectangles come in from the side or bottom, and can be safely shrunk, such that the points no longer covered by these shrunk rectangles are now covered by r'_2.

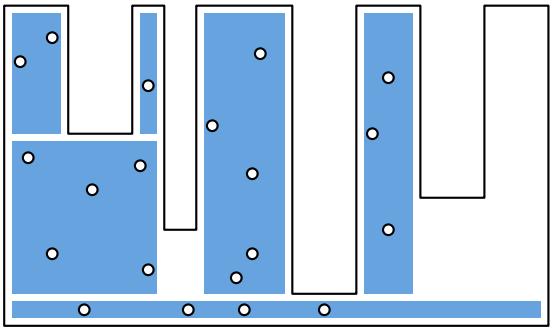


Figure 4 A complete solution for the example in Figure 3, as found by algorithm $\mathcal{A}$.

Finally, we can conclude that $|OPT| = |OPT'|$, since the number of rectangles stays equal in the transformation from OPT to OPT'. After applying this transformation for as long as there are two rectangles in the above horizontally neighboring position, all rectangles correspond to the greedily chosen rectangles of algorithm $\mathcal{A}$. ◀

Finally we show that algorithm $\mathcal{A}$ has polynomial running time.

▶ **Lemma 3.2.** *Algorithm $\mathcal{A}$ runs in $O(n \log n + m \log m)$ time.*

Proof. The $\frac{m}{2}$ horizontal segments of H and the n points in P are separately sorted and put in queues in $O(m \log m)$ and $O(n \log n)$ time, respectively. These queues are then queried, but no additional elements are added, again leading to $O(m \log m)$ and $O(n \log n)$ time spent. Finally, segments of histogram H and solution $\mathcal{S}$ are stored in and removed from status s. Since histogram H has $\frac{m}{2}$ vertical segments, and for each of the $\frac{m}{2}$ horizontal segments of H, only a single rectangle can exist in the solution, status s contains at most $\frac{m}{2}$ segments. As status s is sorted on x-coordinates, we can use a balanced binary search tree to ensure $O(\log m)$ update time. There are $\frac{m}{2}$ events, one for each horizontal segment, and hence maintaining the status takes $O(m \log m)$ time in total. ◀

▶ **Theorem 3.3.** *Given a set P of n points inside a histogram H with m edges, there exists an algorithm that solves the DBCR problem optimally in $O(n \log n + m \log m)$ time.*

References

1. Pankaj K. Agarwal and Jiangwei Pan. Near-linear algorithms for geometric hitting sets and set covers. *Discret. Comput. Geom.*, 63(2):460–482, 2020. doi:10.1007/s00454-019-00099-6.
2. Hee-Kap Ahn, Sang Won Bae, Erik D. Demaine, Martin L. Demaine, Sang-Sub Kim, Matias Korman, Iris Reinbacher, and Wanbin Son. Covering points by disjoint boxes with outliers. *Comput. Geom.*, 44(3):178–190, 2011. doi:10.1016/j.comgeo.2010.10.002.
3. Boris Aronov, Esther Ezra, and Micha Sharir. Small-size ϵ-nets for axis-parallel rectangles and boxes. *SIAM J. Comput.*, 39(7):3248–3282, 2010. doi:10.1137/090762968.
4. Pradeesha Ashok, Sudeshna Kolay, Neeldhara Misra, and Saket Saurabh. Unique covering problems with geometric sets. In Dachuan Xu, Donglei Du, and Dingzhu Du, editors, *Computing and Combinatorics*, pages 548–558, Cham, 2015. Springer International Publishing. doi:10.1007/978-3-319-21398-9_43.
5. Pradeesha Ashok, Sudeshna Kolay, and Saket Saurabh. Multivariate complexity analysis of geometric red blue set cover. *Algorithmica*, 79(3):667–697, 2017. doi:10.1007/s00453-016-0216-x.
6. Pradeesha Ashok, Aniket Basu Roy, and Sathish Govindarajan. Local search strikes again: PTAS for variants of geometric covering and packing. *J. Comb. Optim.*, 39(2):618–635, 2020. doi:10.1007/s10878-019-00432-y.
7. Aritra Banik, Fahad Panolan, Venkatesh Raman, Vibha Sahlot, and Saket Saurabh. Parameterized complexity of geometric covering problems having conflicts. *Algorithmica*, 82(1):1–19, 2020. doi:10.1007/s00453-019-00600-w.
8. Sergey Bereg, Sergio Cabello, José Miguel Díaz-Báñez, Pablo Pérez-Lantero, Carlos Seara, and Inmaculada Ventura. The class cover problem with boxes. *Computational Geometry*, 45(7):294–304, 2012. doi:10.1016/j.comgeo.2012.01.014.
9. Kevin Buchin, Daan Creemers, Andrea Lazzarotto, Bettina Speckmann, and Jules Wulms. Geo word clouds. In *Proc. 2016 IEEE Pacific Visualization Symposium (PacificVis)*, pages 144–151. IEEE Computer Society, 2016. doi:10.1109/PACIFICVIS.2016.7465262.

10 Robert D. Carr, Srinivas Doddi, Goran Konjevod, and Madhav Marathe. On the red-blue set cover problem. In *Proc. 11th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, SODA '00, page 345–353, USA, 2000. Society for Industrial and Applied Mathematics. doi:10.5555/338219.338271.

11 Timothy M. Chan and Nan Hu. Geometric red-blue set cover for unit squares and related problems. *Comput. Geom.*, 48(5):380–385, 2015. doi:10.1016/j.comgeo.2014.12.005.

12 Kenneth L. Clarkson. Algorithms for polytope covering and approximation. In *Proc. 3rd Workshop on Algorithms and Data Structures (WADS)*, volume 709 of *Lecture Notes in Computer Science*, pages 246–252. Springer, 1993. doi:10.1007/3-540-57155-8_252.

13 Kenneth L. Clarkson and Kasturi R. Varadarajan. Improved approximation algorithms for geometric set cover. *Discret. Comput. Geom.*, 37(1):43–58, 2007. doi:10.1007/s00454-006-1273-8.

14 Erik D. Demaine, Uriel Feige, MohammadTaghi Hajiaghayi, and Mohammad R. Salavatipour. Combination can be hard: Approximability of the unique coverage problem. *SIAM J. Comput.*, 38(4):1464–1483, 2008. doi:10.1137/060656048.

15 Herbert Edelsbrunner, Joseph O'Rourke, and Emmerich Welzl. Stationing guards in rectilinear art galleries. *Computer Vision, Graphics, and Image Processing*, 27(2):167 – 176, 1984. doi:10.1016/S0734-189X(84)80041-9.

16 Robert J. Fowler, Mike Paterson, and Steven L. Tanimoto. Optimal packing and covering in the plane are NP-complete. *Inf. Process. Lett.*, 12(3):133–137, 1981. doi:10.1016/0020-0190(81)90111-3.

17 Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. WH Freeman and Company, New York, USA, 1990. doi:10.5555/574848.

18 W. T. Liou, Jimmy J. M. Tan, and Richard C. T. Lee. Minimum partitioning simple rectilinear polygons in $O(n \log \log n)$ time. In *Proc. 5th Symposium on Computational Geometry (SoCG)*, pages 344–353. ACM, 1989. doi:10.1145/73833.73871.

19 Leslie G Valiant. Universality considerations in VLSI circuits. *IEEE Transactions on Computers*, 100(2):135–140, 1981. doi:10.1109/TC.1981.6312176.

Uncertain Curve Simplification

Kevin Buchin[1], Maarten Löffler[*2], Aleksandr Popov[†1], and Marcel Roeloffzen[‡1]

1 **Department of Mathematics and Computer Science, TU Eindhoven, Netherlands**
{k.a.buchin, a.popov, m.j.m.roeloffzen}@tue.nl

2 **Department of Information and Computing Sciences, Utrecht University, Netherlands**
m.loffler@uu.nl

Abstract

We study polygonal curve simplification under uncertainty, where instead of a sequence of exact points, each uncertain point is represented by a region, which contains the (unknown) true location of the vertex. The regions we consider are discrete sets of points, line segments, and convex polygons. We are interested in finding the shortest subsequence of uncertain points such that no matter what the true location of each point is, the resulting polygonal curve is a valid simplification of the original curve under the Hausdorff distance. We present polynomial-time algorithms for this problem.

Related Version An extended version with complete proofs, in which we also discuss the uncertainty modelled with disks and show the set of results for the Fréchet distance, is available on arXiv: *Full Version*: https://arxiv.org/abs/2103.09223 [6]

1 Introduction

Curve simplification is the problem of replacing a polygonal curve by another one which has a similar shape, but fewer vertices. Classical solutions [1], such as those by Ramer and by Douglas and Peucker [7, 15] and by Imai and Iri [8], constrain the new curve to use a subset of the vertices of the old curve and use the Hausdorff distance to measure similarity. Curve simplification is still an active field of study [2, 3, 5, 16].

Typically, it is assumed that the locations of the input curve vertices are known precisely, which is often not the case in real-life data, for instance, when locations are measured using GPS technology. There have been some advances in the study of uncertainty in computational geometry [9, 10, 11, 12, 13], and more recently of uncertain curves [4, 14]. However, to our knowledge, there is no such previous work tackling curve simplification under uncertainty.

An *uncertain curve* consists of a sequence of uncertain points, each modelled as a convex polygon, a line segment, or a discrete set of points that contains the *true* location of the point. When faced with the task of simplifying an uncertain curve, one must consider what is the expected output. We investigate the following practical point of view: we wish to select a subsequence of the (uncertain) input points which is *guaranteed* to be a valid simplification; see Figure 1. Our results are based on adapting the approach by Imai and Iri [8]: we construct a *shortcut graph* which contains all shortcuts that are guaranteed to be valid. In this abstract, we show that we can check a single shortcut in Section 4, using the procedure of Section 3 for several pairs of points.

* Partially supported by the Dutch Research Council (NWO) under project no. 614.001.504.
† Supported by the Dutch Research Council (NWO) under project no. 612.001.801.
‡ Supported by the Dutch Research Council (NWO) under project no. 628.011.005.

37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021.
This is an extended abstract of a presentation given at EuroCG'21. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

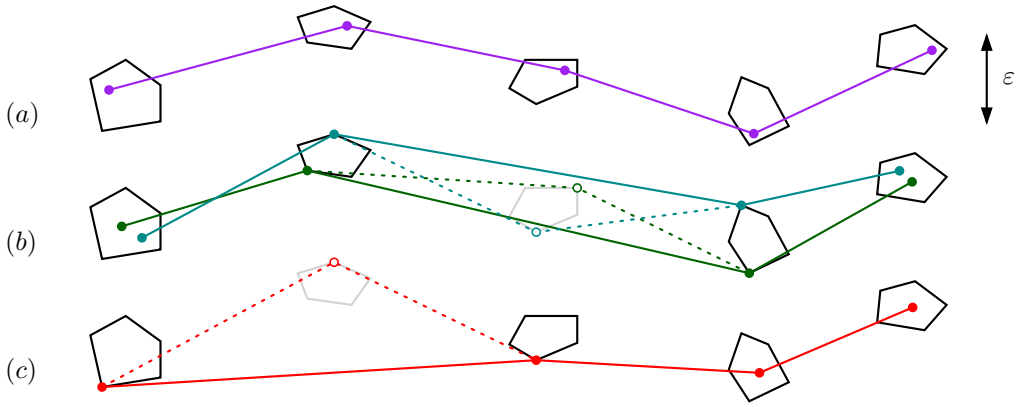


Figure 1 (*a*) An uncertain curve and a potential realisation. (*b*) A valid simplification: for every realisation, the subsequence is within Hausdorff distance ε from the full sequence. (*c*) An invalid simplification: there is a realisation for which the subsequence is not within Hausdorff distance ε.

▶ **Theorem 1.** *We can find the shortest vertex-constrained simplification of an uncertain curve modelled with convex polygons or discrete sets of points, such that for any realisation the simplification is valid under the Hausdorff distance, in time $\mathcal{O}(n^3k^3)$, where k is the size of the point sets or the complexity of the polygons and n is the length of the input curve; and in time $\mathcal{O}(n^3)$ for line segments.*

2 Preliminaries

Denote[1] $[n] \stackrel{\text{def}}{=} \{1, 2, \ldots, n\}$ for any $n \in \mathbb{N}^{>0}$. Given two points $p, q \in \mathbb{R}^2$, denote their Euclidean distance with $\|p - q\|$. Denote a *sequence* of points in $\mathbb{R}^2$ with $\pi = \langle p_1, \ldots, p_n \rangle$. For only two points $p, q \in \mathbb{R}^2$, we also use pq instead of $\langle p, q \rangle$. Denote a subsequence of a sequence π from index i to j with $\pi[i : j] = \langle p_i, p_{i+1}, \ldots, p_j \rangle$. This notation can also be applied if we interpret π as a *polygonal curve* on n vertices (of *length* n). It is defined by linearly interpolating between the successive points in the sequence and can be seen as a continuous function, for $i \in [n-1]$ and $\alpha \in [0, 1]$: $\pi(i + \alpha) = (1 - \alpha)p_i + \alpha p_{i+1}$.

We introduce the notation for order along a curve. Let $p \coloneqq \pi(a)$ and $q \coloneqq \pi(b)$ for some $a, b \in [1, n]$. Then $p \prec q$ iff $a < b$, $p \preccurlyeq q$ iff $a \leq b$, and $p \equiv q$ iff $a = b$.[2] Finally, given points $p, q, r \in \mathbb{R}^2$, define the distance from p to the segment qr as $d(p, qr) \stackrel{\text{def}}{=} \min_{t \in qr} \|p - t\|$.

An *uncertainty region* $U \subset \mathbb{R}^2$ describes a possible location of a point: it has to be inside the region, but we do not know where. Call a sequence of uncertainty regions an *uncertain curve:* $\mathcal{U} = \langle U_1, \ldots, U_n \rangle$. Picking a point from each uncertainty region of $\mathcal{U}$, we get a polygonal curve π called a *realisation* of $\mathcal{U}$, denoted $\pi \Subset \mathcal{U}$. That is, if for some $n \in \mathbb{N}^{>0}$ we have $\pi = \langle p_1, \ldots, p_n \rangle$ and $\mathcal{U} = \langle U_1, \ldots, U_n \rangle$, then $\pi \Subset \mathcal{U}$ if and only if $p_i \in U_i$ for all $i \in [n]$.

Suppose we are given a polygonal curve $\pi = \langle p_1, \ldots, p_n \rangle$, a threshold $\varepsilon \geq 0$, and a curve built on the subsequence of vertices of π for some set $I = \{i_1, \ldots, i_\ell\} \subseteq [n]$, i.e. $\sigma = \langle p_{i_1}, \ldots, p_{i_\ell} \rangle$ with $i_j < i_{j+1}$ for all $j \in [\ell - 1]$ and $\ell \leq n$. We call σ an *ε-simplification* of π if for each segment $\langle p_{i_j}, p_{i_{j+1}} \rangle$, we have $\delta(\langle p_{i_j}, p_{i_{j+1}} \rangle, \pi[i_j : i_{j+1}]) \leq \varepsilon$, where δ denotes some distance measure, e.g. the Hausdorff distance d_{H}.

---

[1] We use $\coloneqq$ and $=\!:$ to denote assignment, $\stackrel{\text{def}}{=}$ for equivalent quantities in definitions or to point out equality by earlier definition, and $=$ in other contexts. We also use $\equiv$, but its usage is always explained.

[2] Note that we can have $p = q$ for $a \neq b$ if the curve intersects itself.

Define a *polygonal closed convex set (PCCS)* as a closed convex set with bounded area that can be described as the intersection of a *finite* number of closed half-spaces. Note that this includes both convex polygons and line segments. Given a PCCS U, let $V(U)$ denote the set of vertices of U. An *indecisive point* is a discrete set of points: $U = \{p^1, \ldots, p^k\}$, with $k \in \mathbb{N}^{>0}$ and $p^i \in \mathbb{R}^2$ for all $i \in [k]$. We solve the following problem to check each shortcut.

▶ **Problem 2.** Given an uncertain curve $\mathcal{U} = \langle U_1, \ldots, U_n \rangle$ on $n \in \mathbb{N}^{\geq 3}$ with $U_i \subset \mathbb{R}^2$ for all $i \in [n]$ modelled as PCCSs or indecisive points, find the maximum possible distance between the curve and its one-segment simplification for any realisation, i.e. find $\max_{\pi \Subset \mathcal{U}} d_{\mathrm{H}}(\pi, \langle \pi(1), \pi(n) \rangle)$.

3 Shortcut Testing: Intermediate Points

In this section, we start with the start and end points fixed, so we are given some $p_1 \in U_1$ and $p_n \in U_n$ and consider realisations $\pi \Subset \mathcal{U}$ with $\pi(1) \equiv p_1$ and $\pi(n) \equiv p_n$. We make use of the following intuitive statements.

▶ **Lemma 3.** *Given four points $a, b, c, d \in \mathbb{R}^2$ forming segments ab and cd, the largest distance from one segment to the other is achieved at an endpoint:*

$$\max_{p \in ab} d(p, cd) = \max \left\{ d(a, cd), d(b, cd) \right\}.$$

▶ **Lemma 4.** *Given $n \in \mathbb{N}^{>0}$, for any precise curve $\pi = \langle p_1, \ldots, p_n \rangle$ with $p_i \in \mathbb{R}^2$ for all $i \in [n]$, we have $d_{\mathrm{H}}(\pi, p_1 p_n) = \max_{i \in [n]} d(p_i, p_1 p_n)$.*

We can show the following lemma, allowing us to test the shortcut with fixed realisations of the endpoints of length n in time $\mathcal{O}(nk)$ for PCCSs with at most k vertices. We can obtain the same equality for indecisive points, replacing $V(U_i)$ with U_i.

▶ **Lemma 5.** *Given $n \in \mathbb{N}^{\geq 3}$, for any uncertain curve modelled with PCCSs $\mathcal{U} = \langle U_1, \ldots, U_n \rangle$ with $U_i \subset \mathbb{R}^2$ for all $i \in [n]$ and $V(U_i) = \{p_i^1, \ldots, p_i^k\}$ for all $i \in [n]$, $k \in \mathbb{N}^{>0}$, and given some $p_1 \in U_1$ and $p_n \in U_n$, we have*

$$\max_{\pi \Subset \mathcal{U}, \pi(1) \equiv p_1, \pi(n) \equiv p_n} d_{\mathrm{H}}(\pi, p_1 p_n) = \max_{i \in \{2, \ldots, n-1\}} \max_{v \in V(U_i)} d(v, p_1 p_n).$$

Proof. Assume the setting of the lemma. Derive $\max_{\pi \Subset \mathcal{U}, \pi(1) \equiv p_1, \pi(n) \equiv p_n} d_{\mathrm{H}}(\pi, p_1 p_n) = \max_{\pi \Subset \mathcal{U}, \pi(1) \equiv p_1, \pi(n) \equiv p_n} \max_{i \in [n]} d(\pi(i), p_1 p_n) = \max_{i \in \{2, \ldots, n-1\}} \max_{p \in U_i} d(p, p_1 p_n)$, where the steps hold by Lemma 4 and the definition of $\Subset$. It is trivial to show that for any PCCS U and a line segment ab it holds that $\max_{p \in U} d(p, ab) = \max_{v \in V(U)} d(v, ab)$. ◀

4 Shortcut Testing: All Points

In the previous section, we have covered testing a shortcut, given that the first and last points are fixed. Here we generalise the problem: we check if $\max_{\pi \Subset \mathcal{U}} d_{\mathrm{H}}(\pi, \langle \pi(1), \pi(n) \rangle) \leq \varepsilon$.

First of all, for indecisive curves, we can just use the approach of the previous section for each pair of realisations from $U_1 \times U_n$. This way we test a single shortcut in time $\mathcal{O}(nk^3)$.

4.1 Non-intersecting Polygonal Closed Convex Sets

Consider first the case where the interiors of convex polygons U_1 and U_n do not intersect.

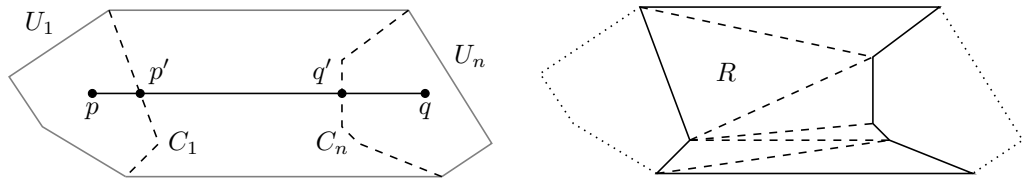


Figure 2 Left: Illustration for Observation 6. The convex hull of the disks is shown in grey. The dotted chains are C_1 and C_n. Any line segment pq with $p \in U_1$ and $q \in U_n$ crosses C_1 and C_n. Right: Illustration for the procedure. The region R is triangulated.

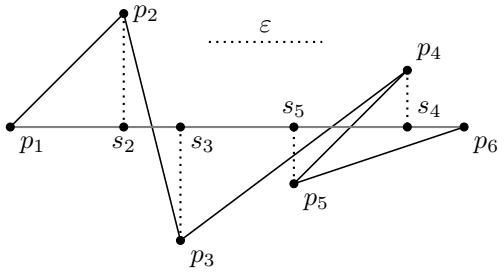


Figure 3 Alignment for the Hausdorff distance, described as $\langle s_1 \coloneqq p_1, s_2, s_3, s_4, s_5, s_6 \coloneqq p_6 \rangle$.

▶ **Observation 6.** Given an uncertain curve modelled by convex polygons $\mathcal{U} = \langle U_1, \ldots, U_n \rangle$ with the interiors of U_1 and U_n not intersecting, note:

- There are two *outer tangents* to the polygons U_1 and U_n, and the convex hull of $U_1 \cup U_n$ consists of a convex chain from U_1, a convex chain from U_n, and the outer tangents.
- Let C_i be the convex chain from U_i that is not part of the convex hull for $i \in \{1, n\}$; then

$$\max_{\pi \in \mathcal{U}} d_{\mathrm{H}}(\pi, \langle \pi(1), \pi(n) \rangle) \leq \varepsilon \iff \max_{\pi \in \mathcal{U}, \pi(1) \in C_1, \pi(n) \in C_n} d_{\mathrm{H}}(\pi, \langle \pi(1), \pi(n) \rangle) \leq \varepsilon \,.$$

To see that the second observation is true, note that one direction is trivial. In the other direction, note that any line segment pq with $p \in U_1$, $q \in U_n$ crosses both C_1 and C_n, say, at $p' \in C_1$ and $q' \in C_n$. We know that there is a valid alignment for $p'q'$, both for the Hausdorff and the Fréchet distance; we can then use this alignment for pq. See Figure 2.

We claim that we can use the following procedure to check $\max_{\pi \in \mathcal{U}} d_{\mathrm{H}}(\pi, \langle \pi(1), \pi(n) \rangle) \leq \varepsilon$.

1. Triangulate the region R bounded by two convex chains C_1 and C_n and the outer tangents.
2. Check that $\max_{\pi \in \mathcal{U}, \pi(1) \equiv s, \pi(n) \equiv t} d_{\mathrm{H}}(\pi, st) \leq \varepsilon$ for each line segment st of the triangulation with $s \in C_1$, $t \in C_n$.

First of all, observe that we can compute a triangulation (see Figure 2), and that every triangle has two points from one convex chain and one point from the other chain. If all three points were from the same chain, then the triangle would lie outside of R. Now consider some line segment pq with $p \in C_1$, $q \in C_n$. To complete the argument, it remains to show that the checks in step 2 mean that also $\max_{\pi \in \mathcal{U}, \pi(1) \equiv p, \pi(n) \equiv q} d_{\mathrm{H}}(\pi, pq) \leq \varepsilon$. Observe that the triangles span across the region R, so when going from one tangent to the other within R we cross all the triangles. Therefore, we can number the edges of the triangles that go from C_1 to C_n, in the order of occurrence on such a path, from 1 to k. We can establish a Hausdorff *alignment*, as shown intuitively in Figure 3; denote the alignment established on line $j \in [k]$ with the sequence of a_i^j, for $i \in \{2, \ldots, n-1\}$. We can establish polygonal curves $A_i \coloneqq \langle a_i^1, \ldots, a_i^k \rangle$; they all stay within R (see Figure 4). We claim that for any line segment pq defined above, we can establish a valid alignment from intersection points of pq and A_i.

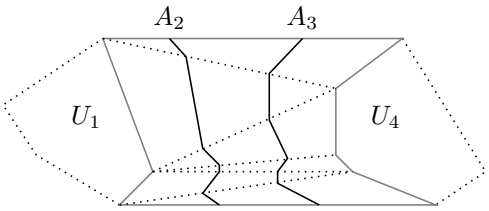


Figure 4 An example set of curves $A = \{A_2, A_3\}$ discussed in Lemma 7.

▶ **Lemma 7.** *Given a set of curves $A := \{A_2, \ldots, A_{n-1}\}$ in R described above and a line segment pq with $p \in C_1$, $q \in C_n$, we have $\max_{\pi \in \mathcal{U}, \pi(1) \equiv p, \pi(n) \equiv q} d_{\mathrm{H}}(\pi, pq) \leq \varepsilon$.*

Proof. Note that pq crosses each A_i at least once. We can take any one crossing for each i and establish the alignment. Consider such a crossing point p'_i. It falls in some triangle bounded by a segment from either C_1 or C_n and two line segments that contain points a_i^j and a_i^{j+1} for some $j \in [k-1]$. We know, using Lemma 5, that $\max_{w \in U_i} \|a_i^j - w\| \leq \varepsilon$ and $\max_{w \in U_i} \|a_i^{j+1} - w\| \leq \varepsilon$. Consider any point $w' \in U_i$. Then, using Lemma 3 with $c := d := w'$, we find that $\|w' - p'_i\| \leq \varepsilon$. Therefore, also $\max_{w \in U_i} \|p'_i - w\| \leq \varepsilon$; using Lemma 5, we conclude that indeed $\max_{\pi \in \mathcal{U}, \pi(1) \equiv p, \pi(n) \equiv q} d_{\mathrm{H}}(\pi, pq) \leq \varepsilon$. ◀

The proof of Lemma 7 shows how to solve the problem for two convex polygons with non-intersecting interiors. We can also use it directly for the case of line segments that do not intersect except at endpoints. Furthermore, in this particular case it is not necessary to use a triangulation, so we can get rid of one of the three resulting line segments.

▶ **Lemma 8.** *Given $n \in \mathbb{N}^{\geq 3}$, for any uncertain curve modelled with line segments $\mathcal{U} = \langle U_1, \ldots, U_n \rangle$ with $U_i = p_i^1 p_i^2 \subset \mathbb{R}^2$ for all $i \in [n]$, given a threshold $\varepsilon \in \mathbb{R}^{>0}$, and given that $U_1 \cap U_n \subset \{p_1^1, p_1^2\}$, and assuming that the triangles $p_1^1 p_n^1 p_1^2$ and $p_1^2 p_n^1 p_n^2$ form a triangulation of the convex hull of $U_1 \cup U_n$, we have $\max_{\pi \in \mathcal{U}} \delta(\pi, \langle \pi(1), \pi(n) \rangle) \leq \varepsilon$ if and only if*

$$\max \Big\{ \max_{\pi \in \mathcal{U}, \pi(1) \equiv p_1^1, \pi(n) \equiv p_n^1} \delta(\pi, p_1^1 p_n^1), \max_{\pi \in \mathcal{U}, \pi(1) \equiv p_1^2, \pi(n) \equiv p_n^2} \delta(\pi, p_1^2 p_n^2) \Big\} \leq \varepsilon \,.$$

4.2 Intersecting Polygonal Closed Convex Sets

We proceed to discuss the situation where the interiors of U_1 and U_n intersect, or where line segments U_1 and U_n cross.

Line segments. Assume line segments $U_1 \stackrel{\text{def}}{=} p_1^1 p_1^2$ and $U_n \stackrel{\text{def}}{=} p_n^1 p_n^2$ cross; call their intersection point s. Then we can use Lemma 8 separately on pairs of $\{p_1^1 s, s p_1^2\} \times \{p_n^1 s, s p_n^2\}$. Clearly, together this will cover the entire set of realisations of pq with $p \in U_1$, $q \in U_n$.

▶ **Lemma 9.** *Given $n \in \mathbb{N}^{\geq 3}$, for any uncertain curve modelled with line segments $\mathcal{U} = \langle U_1, \ldots, U_n \rangle$ with $U_i = p_i^1 p_i^2 \subset \mathbb{R}^2$ for all $i \in [n]$, given a threshold $\varepsilon \in \mathbb{R}^{>0}$, we can check that $\max_{\pi \in \mathcal{U}} d_{\mathrm{H}}(\pi, \langle \pi(1), \pi(n) \rangle) \leq \varepsilon$.*

Convex polygons. Convex polygons whose interiors intersect can be partitioned along the intersection lines, so into a convex polygon $R := U_1 \cap U_n$ and two sets of polygons $\mathcal{P}_1 := \{P_1^1, \ldots, P_1^k\}$ and $\mathcal{P}_n := \{P_n^1, \ldots, P_n^\ell\}$ for some $k, \ell \in \mathbb{N}^{>0}$. We can look at pairs from $\mathcal{P}_1 \times \mathcal{P}_n$ separately. The pairs where R is involved are treated later. Consider some $(P, Q) \in \mathcal{P}_1 \times \mathcal{P}_n$. Note that P and Q are convex polygons with a convex cut-out, so the

boundary forms a convex chain, followed by a concave chain. We need to compute some convex polygons P' and Q' with non-intersecting interiors that are equivalent to P and Q, so that we can apply the approaches from Section 4.1.

We claim that we can simply take the convex hull of P and Q to obtain P' and Q'. Clearly, the resulting polygons will be convex. Also, the concave chains of P are bounded by points s and t and are replaced with the line segment st; same happens for Q with point u and v. The points s, t, u, v are points of intersection of original polygons U_1 and U_n, so they lie on the boundary of R, and their order along that boundary can only be s, t, u, v or s, t, v, u. Thus, it cannot happen that st crosses uv, and it cannot be that uv is in the interior of the convex hull of P, as otherwise R would not be convex. Hence, the interiors of P' and Q' cannot intersect, so they satisfy the necessary conditions.

Finally, we need to show that the solution for (P', Q') is equivalent to that for (P, Q). One direction is trivial, as $P \subseteq P'$ and $Q \subseteq Q'$; for the other direction, consider any line segment that leaves P through the concave chain. In our approach, we test the lines starting in s and t; the established alignments are connected into paths. The paths A_i do not cross st. So, any alignment in the region of $\mathrm{CH}(P \cup Q) \setminus (P \cup Q)$ can also be made in the region $\mathrm{CH}(P' \cup Q') \setminus (P' \cup Q')$. So, this approach yields valid solutions for all pairs not involving R.

Now consider the pair (R, R). A curve may now consist of a single point, so all the points of U_i need to be close enough to all the points of R. To check that, observe that the pair of points $p \in U_i$ and $q \in R$ that has maximal distance has the property that p is an extreme point of U_i in direction qp and q is an extreme point of R in direction pq. So, it suffices, starting at the rightmost point of U_i and leftmost point of R in some coordinate system, to then rotate clockwise around both regions keeping track of the distance between tangent points. Note that only vertices need to be considered, as the extremal point cannot lie on an edge. Finally, any other pair that involves R is covered by the stronger case of (R, R): for any line we can align every intermediate object with any point in R.

▶ **Lemma 10.** *Given $n \in \mathbb{N}^{\geq 3}$, for any uncertain curve modelled with convex polygons $\mathcal{U} = \langle U_1, \ldots, U_n \rangle$ with $U_i \subset \mathbb{R}^2$ for all $i \in [n]$ and $V(U_i) = \{p_i^1, \ldots, p_i^k\}$ for all $i \in [n]$, $k \in \mathbb{N}^{>0}$, given a threshold $\varepsilon \in \mathbb{R}^{>0}$, we can check that $\max_{\pi \in \mathcal{U}} d_{\mathrm{H}}(\pi, \langle \pi(1), \pi(n) \rangle) \leq \varepsilon$.*

5 Combining Steps

In the previous sections, we have shown how to check if a shortcut of length $n \geq 3$ is valid under the Hausdorff distance, for indecisive points and polygonal closed convex sets. It is easy to see that a shortcut of length $n = 2$ is always valid. Therefore, we can use the previously described procedures to construct a shortcut graph; any path in such a graph from the vertex 1 to vertex n corresponds to a valid simplification, so the shortest path gives us the result we need. The graph has $\mathcal{O}(n^2)$ edges. These observations lead to Theorem 1.

References

1 Pankaj K. Agarwal and Kasturi R. Varadarajan. Efficient algorithms for approximating polygonal chains. *Discrete & Computational Geometry*, 23(2):273–291, 2000. doi:10.1007/PL00009500.

2 Gill Barequet, Danny Z. Chen, Ovidiu Daescu, Michael T. Goodrich, and Jack S. Snoeyink. Efficiently approximating polygonal paths in three and higher dimensions. *Algorithmica*, 33(2):150–167, 2002. doi:10.1007/s00453-001-0096-5.

3 Karl Bringmann and Bhaskar Ray Chaudhury. Polyline simplification has cubic complexity. In *35th International Symposium on Computational Geometry (SoCG 2019)*, volume 129

of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 18:1–18:16, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2019.18.

4 Kevin Buchin, Chenglin Fan, Maarten Löffler, Aleksandr Popov, Benjamin Raichel, and Marcel Roeloffzen. Fréchet distance for uncertain curves. In *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*, volume 168 of *LIPIcs*, pages 20:1–20:20, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2020.20.

5 Kevin Buchin, Maximilian Konzack, and Wim Reddingius. Progressive simplification of polygonal curves. *Computational Geometry*, 88:101620:1–101620:18, 2020. doi:10.1016/j.comgeo.2020.101620.

6 Kevin Buchin, Maarten Löffler, Aleksandr Popov, and Marcel Roeloffzen. Uncertain curve simplification, March 2021. arXiv:2103.09223.

7 David H. Douglas and Thomas K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973. doi:10.3138/FM57-6770-U75U-7727.

8 Hiroshi Imai and Masao Iri. Computational-geometric methods for polygonal approximations of a curve. *Computer Vision, Graphics, and Image Processing*, 36(1):31–41, 1986. doi:10.1016/S0734-189X(86)80027-5.

9 Allan Jørgensen, Jeff M. Phillips, and Maarten Löffler. Geometric computations on indecisive points. In *Algorithms and Data Structures (WADS 2011)*, volume 6844 of *Lecture Notes in Computer Science*, pages 536–547, Berlin, Germany, 2011. Springer Berlin Heidelberg. doi:10.1007/978-3-642-22300-6_45.

10 Christian Knauer, Maarten Löffler, Marc Scherfenberg, and Thomas Wolle. The directed Hausdorff distance between imprecise point sets. *Theoretical Computer Science*, 412(32):4173–4186, 2011. doi:10.1016/j.tcs.2011.01.039.

11 Maarten Löffler. *Data Imprecision in Computational Geometry*. PhD thesis, Universiteit Utrecht, October 2009. URL: https://dspace.library.uu.nl/bitstream/handle/1874/36022/loffler.pdf [cited 2019-06-15].

12 Maarten Löffler and Jeff M. Phillips. Shape fitting on point sets with probability distributions: ESA 2009. In *Algorithms*, number 5757 in LNCS, pages 313–324, Berlin, Germany, 2009. Springer Berlin Heidelberg. arXiv:0812.2967v1, doi:10.1007/978-3-642-04128-0_29.

13 Maarten Löffler and Marc van Kreveld. Largest and smallest tours and convex hulls for imprecise points. In *Algorithm Theory – SWAT 2006*, volume 4059 of *Lecture Notes in Computer Science*, pages 375–387, Berlin, Germany, 2006. Springer Berlin Heidelberg. doi:10.1007/11785293_35.

14 Aleksandr Popov. Similarity of uncertain trajectories. Master's thesis, Eindhoven University of Technology, November 2019. URL: https://research.tue.nl/en/studentTheses/similarity-of-uncertain-trajectories [cited 2019-12-18].

15 Urs Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*, 1(3):244–256, 1972. doi:10.1016/S0146-664X(72)80017-0.

16 Mees van de Kerkhof, Irina Kostitsyna, Maarten Löffler, Majid Mirzanezhad, and Carola Wenk. Global curve simplification. In *27th Annual European Symposium on Algorithms (ESA 2019)*, volume 144 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 67:1–67:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ESA.2019.67.