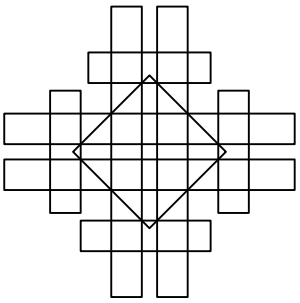
34th International Symposium on Computational Geometry

SoCG 2018, June 11–14, 2018, Budapest, Hungary

Edited by

Bettina Speckmann

Csaba D. Tóth



LIPICS

Editors

Bettina Speckmann	Csaba D. Tóth
TU Eindhoven	Cal State Northridge
Eindhoven	Los Angeles, CA
The Netherlands	USA
b.speckmann@tue.nl	csaba.toth@csun.edu

ACM Classification 2012
Theory of computation → Computational geometry, Mathematics of computing → Combinatorics, Theory of computation → Design and analysis of algorithms

ISBN 978-3-95977-066-8

Published online and open access by
Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at http://www.dagstuhl.de/dagpub/978-3-95977-066-8.

Publication date
June 2018

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at http://dnb.d-nb.de.

Digital Object Identifier: 10.4230/LIPIcs.SoCG.2018.0

LIPIcs – Leibniz International Proceedings in Informatics

LIPIcs is a series of high-quality conference proceedings across all fields in informatics. LIPIcs volumes are published according to the principle of Open Access, i.e., they are available online and free of charge.

ISSN 1868-8969

http://www.dagstuhl.de/lipics

Contents

Papers

34th International Symposium on Computational Geometry (SoCG 2018).
Editors: Bettina Speckmann and Csaba D. Tóth
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Multimedia Exposition

Foreword

The 34th International Symposium on Computational Geometry (SoCG) will be held in Budapest, Hungary, June 11–14, 2018, as part of the Computational Geometry Week. A record number of 206 papers have been submitted to SoCG 2018. After a thorough review process, in which each paper has been evaluated by three or more independent reviewers, the Program Committee accepted 73 papers for presentation at SoCG. These proceedings contain extended abstracts of the accepted papers, limited to 13 pages plus references. If any supporting material (e.g., proofs or experimental details) does not fit in the page limit, the full paper is available at a public repository, which is referenced in the extended abstract.

The Best Paper Award goes to the paper "Shellability is NP-complete" by Xavier Goaoc, Pavel Paták, Zuzana Patáková, Martin Tancer, and Uli Wagner. The Best Student Presentation Award will be determined and announced at the symposium, based on ballots cast by the attendees.

A selection of papers, recommended by the Program Committee, have been invited to forthcoming special issues of *Discrete & Computational Geometry* and the *Journal of Computational Geometry*, dedicated to the best papers of the symposium.

In addition to the technical papers, there were five submissions to the multimedia exposition (three videos and two applets). All five were reviewed and four were accepted for presentation. The extended abstracts that describe these submissions are included in this proceedings volume. The multimedia content can be found at `http://www.computational-geometry.org`.

We thank the authors of all submitted papers and multimedia presentations. We are grateful to the members of the SoCG Program Committee, the Multimedia Committee, and 340 additional reviewers for their dedication and expertise that ensure the high quality of the papers in these proceedings. We would also like to thank the Proceedings Chair, Wouter Meulemans, for his meticulous work preparing the final proceedings. Many other people contributed to the success of SoCG 2018 and the entire CG Week. We especially thank the local organizers, all members of the Workshop and YRF Committees, and the Computational Geometry Steering Committee.

Bettina Speckmann
Program Committee, co-chair
TU Eindhoven

Csaba D. Tóth
Program Committee, co-chair
Cal State Northridge

Xavier Goaoc
Multimedia Committee, chair
Université Paris-Est Marne-la-Vallée

34th International Symposium on Computational Geometry (SoCG 2018).
Editors: Bettina Speckmann and Csaba D. Tóth
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Conference Organization

SoCG Program Committee

Luis Barba *(ETH Zürich, Switzerland)*
Pavle V.M. Blagojević *(FU Berlin, Germany, and Mathematical Institute of SASA, Serbia)*
Karl Bringmann *(Max-Planck-Institut für Informatik, Germany)*
Siu-Wing Cheng *(Hong Kong University of Science and Technology, China)*
Khaled Elbassioni *(Masdar Institute, United Arab Emirates)*
Jeff Erickson *(University of Illinois at Urbana-Champaign, USA)*
Fabrizio Frati *(Roma Tre University, Italy)*
Jie Gao *(Stony Brook University, USA)*
Andreas Holmsen *(KAIST, Republic of South Korea)*
Minghui Jiang *(Utah State University, USA)*
Michael Kerber *(TU Graz, Austria)*
David Mount *(University of Maryland, USA)*
Elizabeth Munch *(Michigan State University, USA)*
Steve Oudot *(INRIA Saclay, France)*
Dömötör Pálvölgyi *(Eötvös Loránd University, Hungary)*
Benjamin Raichel (*University of Texas at Dallas, USA)*
Orit E. Raz *(University of British Columbia, Canada)*
Bettina Speckmann *(co-chair, TU Eindhoven, the Netherlands)*
Andrew Suk *(University of California, San Diego, USA)*
Csaba D. Tóth *(co-chair, California State University Northridge and Tufts University, USA)*
Shira Zerbib *(University of Michigan and MSRI, USA)*

SoCG Proceedings Chair

Wouter Meulemans *(TU Eindhoven, the Netherlands)*

Multimedia Program Committee

Satyan Devadoss *(University of San Diego, USA)*
Anne Driemel *(TU Eindhoven, the Netherlands)*
Xavier Goaoc *(chair, Université Paris-Est Marne-la-Vallée, France)*
Francis Lazarus *(CNRS, France)*
Mira Shalah *(Stanford University, USA)*
Frank Staals *(Utrecht University, the Netherlands)*
Ileana Streinu *(Smith College, USA)*

Workshop Program Committee

Maarten Löffler *(Utrecht University, the Netherlands)*
Brittany Terese Fasy *(Montana State University, USA)*
Suresh Venkatasubramanian *(University of Utah, USA)*
Carola Wenk *(chair, Tulane University, USA)*

34th International Symposium on Computational Geometry (SoCG 2018).
Editors: Bettina Speckmann and Csaba D. Tóth
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Young Researchers Forum Program Committee

Peyman Afshani *(Aarhus University, Denmark)*
Chao Chen *(CUNY Queens College, USA)*
Elena Khramtcova *(Université Libre de Bruxelles, Belgium)*
Irina Kostitsyna *(TU Eindhoven, the Netherlands)*
Jon Lenchner *(IBM Research, Africa, USA)*
Nabil Mustafa *(ESIEE Paris, France)*
Amir Nayyeri *(Oregon State University, USA)*
Don Sheehy *(chair, University of Connecticut, USA)*
Haitao Wang *(Utah State University, USA)*
Yusu Wang *(The Ohio State University, USA)*

Local Organizing Committee

Imre Bárány *(Rényi Institute of Mathematics, Hungary, and University College London, UK)*
Balázs Keszegh *(Rényi Institute of Mathematics, Hungary)*
Dezső Miklós *(Rényi Institute of Mathematics, Hungary)*
Márton Naszódi *(Eötvös Loránd University, Hungary)*
János Pach *(chair, Rényi Institute of Mathematics, Hungary, and EPFL, Switzerland)*
Dömötör Pálvölgyi *(Eötvös Loránd University, Hungary)*
Géza Tóth *(Rényi Institute of Mathematics, Hungary)*

Steering Committee (2016-)

Erin Chambers *(Saint Louis University, USA)*
Dan Halperin *(secretary, Tel Aviv University, Israel)*
Marc van Kreveld *(Utrecht University, the Netherlands)*
Joseph S. B. Mitchell *(treasurer, Stony Brook University, USA)*
David Mount *(University of Maryland, USA)*
Monique Teillaud *(chair, INRIA Nancy – Grand Est, France)*

Additional Reviewers

Mohammad Ali Abam
Amir Abboud
Ahmed Abdelkader
Mikkel Abrahamsen
Aditya Acharya
Eyal Ackerman
Michal Adamaszek
Henry Adams
Peyman Afshani
Hee-Kap Ahn
Oswin Aichholzer
Hugo Akitaya
Alexandr Andoni
Patrizio Angelini
Boris Aronov
Sunil Arya
Sergey Avvakumov
Sang Won Bae
Martin Balko
Sayan Bandyapadhyay
Bahareh Banyassady
Jérémy Barbay
Danielle Barnes
Yair Bartal
Abdul Basit
Ulrich Bauer
Michael Bekos
Ioana Bercea
Sergey Bereg
Mark de Berg
Nicolas Berkouk
Therese Biedl
Ahmad Biniaz
Christopher Bishop
Thomas Bläsius
Jean-Daniel Boissonnat
Nicolas Bonichon
Édouard Bonnet
Prosenjit Bose
Magnus Botnan
Cornelius Brand
Mickaël Buchet
Kevin Buchin
Maike Buchin
Boris Bukh
Sergiy Butenko
Sergio Cabello
Mathieu Carrière
Michael Catanzaro
Parinya Chalermsook
Erin Chambers
T.-H. Hubert Chan
Timothy M. Chan
Hsien-Chih Chang
Yi-Jun Chang
John Chaussard
Frédéeric Chazal
Chao Chen
Otfried Cheong
Markus Chimani
Aruni Choudhary
Jérémy Cochoy
Vincent Cohen-Addad
David Cohen-Steiner
René Corbet
Bruno Courcelle
Justin Curry
Guilherme D. Da Fonseca
Giordano Da Lozzo
Gábor Damásdi
Philip Dasler
Jean-Lou De Carufel
Olivier Devillers
Hu Ding
Jiaxin Ding
Vincent Divol
Paweł Dłotko
Michael Gene Dobbins
Anne Driemel
Adrian Dumitrescu
Herbert Edelsbrunner
David Eisenstat
Marek Elias
Ioannis Emiris
Alina Ene
David Eppstein
Esther Ezra
Sándor Fekete
Dan Feldman
Stefan Felsner
Alejandro Flores Velazco
Steven Fortune
Kyle Fox
Dirk Frettlöh
Florian Frick
Sorelle Friedler
Ulderico Fugacci
Radoslav Fulek
Bernd Gärtner
Ellen Gasparovic
Daniel Gerbner
Robert Ghrist
Panos Giannopoulos
Anna Gilbert
Chad Giusti
Marc Glisse
Xavier Goaoc
Chaim Goodman-Strauss
Lee-Ad Gottlieb
Sathish Govindarajan
Kasper Green Larsen
Martin Gronemann
Anupam Gupta
Dan Halperin
Sariel Har-Peled
David Harris
Meng He
Martin Henk
Greg Henselman
John Hershberger
Darryl Hill
Michael Hoffmann
Ivor Hoog V.D.
Ingrid Hotz
Tamas Hubai
Alfredo Hubard
R. Inkulu
Martin Jaggi
Bruno Jartoux
Slobodan Jelić
Ross J. Kang
Iyad Kanj

34th International Symposium on Computational Geometry (SoCG 2018).
Editors: Bettina Speckmann and Csaba D. Tóth
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Haim Kaplan
Roman Karasev
Matthew Katz
Ken-Ichi Kawarabayashi
Mark Keil
Chaya Keller
Balázs Keszegh
Ralph Keusch
Elena Khramtcova
Minki Kim
Guy Kindler
David Kirkpatrick
Sándor Kisfaludi-Bak
Steven Klee
Michal Kleinbort
Matias Korman
Dániel Korándi
Miroslav Kramar
Marc van Kreveld
Sebastian Krinninger
Slava Krushkal
Nirman Kumar
Piyush Kumar
Marvin Künnemann
Andrey Kupavskii
Vitaliy Kurlin
O-Joung Kwon
Jan Kynčl
Jean-Philippe Labbé
Theo Lacombe
Zsolt Lángi
Seunghun Lee
Erik Jan van Leeuwen
Johannes Lengler
Michael Lesnick
David Letscher
Rachel Levanger
Christos Levcopoulos
Bernard Lidicky
Patrick Lin
Chih-Hung Liu
Anna Lubiw
Ben Lund
Alexander Magazinov
Johann Makowsky
Yuchen Mao
Vasileios Maroulas
Leonardo Martínez-Sandoval
Domagoj Matijević
Tyler Mayer
Gregory McColm
Ali Mehrabi
Arnaud de Mesmay
Abhishek Methuku
Frédéric Meunier
Tamás Mezei
Samuel Micka
Tillmann Miltzow
Mojgan Mirzaei
Majid Mirzanezhad
Joseph S. B. Mitchell
Dylan Molho
Debajyoti Mondal
Fabrizio Montecchiani
Shay Moran
Pat Morin
Dmitriy Morozov
Shay Mozes
Wolfgang Mülzer
Nabil Mustafa
Zoltán Nagy
Waleed Najy
Márton Naszódi
Amir Nayyeri
Ofer Neiman
Yakov Nekrich
Eran Nevo
Trung Thanh Nguyen
Arnur Nigmetov
Gabriel Nivasch
Jerri Nummenpalo
André Nusser
Joseph O'Rourke
Eunjin Oh
Yoshio Okamoto
Aurélien Ooms
Tim Ophelders
Arnau Padrol
Rasmus Pagh
Gaiane Panina
Evanthia Papadopoulou
Dimitris Papailiopoulos
Amit Patel
Maurizio Patrignani
Pavel Paták
Zuzana Patáková
Michael Pelsmajer
Richard Peng
Jose Perea
Ljubomir Perkovic
Thang Pham
Jeff Phillips
Alexander Pilz
Rom Pinchasi
Cosmin Pohoata
Alexey Pokrovskiy
Marc Pouget
Lionel Pournin
Eric Price
Siddharth Pritam
Kent Quanrud
Sharath Raghvendra
Rajiv Raman
Bhaskar Ray Chaudhury
Saurabh Ray
Ilya Razenshteyn
André van Renssen
Bruce Richter
Vanessa Robins
Liam Roditty
Marcel Roeloffzen
Edgardo Roldan-Pensado
Vincenzo Roselli
Günter Rote
Alan Roytman
Natan Rubin
Ignaz Rutter
Yogish Sabharwal
Michael Sagraloff
Gelasio Salazar
Francisco Santos
Rik Sarkar
Radmila Sazdanovic
Marcus Schaefer
Jean-Marc Schlenker
Christiane Schmidt
Patrick Schnider
Hannah Schreiber
Matthias Schymura
Don Sheehy
Adam Sheffer
Thomas Shermer
Jonathan Shewchuk
Ilya Shkredov

Anastasios Sidiropoulos
Vin de Silva
Francesco Silvestri
René Sitters
Arkadiy Skopenkov
Mikhail Skopenkov
Primoz Skraba
Michiel Smid
Shakhar Smorodinsky
Pablo Soberon
Israela Solomon
Kiril Solovey
Jonathan Spreer
Frank Staals
Frank van der Stappen
Daniel Štefankovič
Anastasios Stefanou
Miloš Stojaković
Darren Strash
Christopher Sukhu
He Sun
Shuhao Tan
Martin Tancer
Raphael Tinarrage
Hans Raj Tiwary
István Tomon
Christopher Tralie
Konstantinos Tsakalidis
Katharine Turner
Géza Tóth
Torsten Ueckerdt
Jérôme Urhausen
Greg Van Buskirk
Kasturi Varadarajan
Mikael Vejdemo-Johansson
Kevin Verbeek
Sander Verdonschot
Antoine Vigneron
Máté Vizer
Hubert Wagner
Uli Wagner
Bartosz Walczak
Haitao Wang
Shiyu Wang
Weifu Wang
Yusu Wang
Oren Weimann
Shmuel Weinberger
Rene Weller
Mathijs Wintraecken
Ching Wong
Matthew Wright
Christian Wulff-Nilsen
Ge Xia
Jinhui Xu
Ke Yi
Fang-Yi Yu
Jingjin Yu
Yelena Yuditsky
Joshua Zahl
Alireza Zarei
Frank de Zeeuw
Meirav Zehavi
Ahad N. Zehmakan

Acknowledgement of Support

We are grateful to Rényi Institute of Mathematics for hosting and sponsoring CG Week 2018, and to the Hungarian Academy of Sciences (MTA) and the National Science Foundation (NSF) for their support.

34th International Symposium on Computational Geometry (SoCG 2018).
Editors: Bettina Speckmann and Csaba D. Tóth
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Invited Talk

Stories Are Not Just Words: How Visualization Helps Us to Explain, Reason, Explore and Remember

Jo Wood
giCentre
City, University of London
United Kingdom

Abstract

From Euclid's *Elements* and Liu Hui's *The Sea Island Mathematical Manual* through Decartes and Euler to Mandlebrot and Wolfram, we have always used images to assist telling stories about geometry. The long history of *proof without words* demonstrates that images alone can sometimes tell convincing mathematical stories. In a parallel history we have for millennia used careful geometric projection onto a plane to tell cartographic stories of discoveries made, lands yet visited and battles to be fought. In this talk I explore some of the examples and theory of storytelling using visualization and cartography in order to evaluate whether computational geometers have anything to gain through visual storytelling and whether some problems that persist in visual storytelling might be solvable by computational geometers.

I explore the coupling of textual narrative and visualization to support explanation and reasoning as well as more open-ended exploration by considering the *literate programming* approach advocated by Donald Knuth, the *computational essays* of Stephen Wolfram and the newly emerging paradigm of *literate visualization*. These are most readily seen in notebook environments of data science such as *R-Notebook*, *Jupyter Lab* and most recently *Observable*. I illustrate literate visualization with examples from these environments as well as one newly developed by the giCentre – *litvis*. I argue that visual approaches to computation are valuable as they support a shift from a dialogue between person and computer to one between people.

2012 ACM Subject Classification Human-centered computing → Information visualization

Keywords and phrases Literate visualization, geovisualization, storytelling, literate programming

34th International Symposium on Computational Geometry (SoCG 2018).
Editors: Bettina Speckmann and Csaba D. Tóth

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Invited Talk
Circle Squaring and Other Combinatorial Problems in Geometric Measure Theory

András Máthé
Mathematics Institute
University of Warwick
Coventry, United Kingdom

Abstract

I will survey some results and open problems in geometric measure theory of combinatorial flavour.

The famous Banach-Tarski paradox states that in the three dimensional space a ball of radius one can be partitioned into finitely many (non-measurable) pieces that can be rearranged (applying rotations and translations) to obtain a ball of radius 2. On the other hand, Tarski's circle squaring problem asked if it was possible to partition a disc in the plane into finitely many pieces and rearrange these to obtain the square (of the same area). This was shown to be possible by Laczkovich in 1990. I will talk about the recent results that show that this "circle squaring" is possible by using pieces that are Lebesgue measurable, or even Borel (by Marks and Unger).

I will also mention some results and questions about patterns in fractal sets and a problem about the fractal analogue of the Szemerédi-Trotter theorem of point-line incidences.

2012 ACM Subject Classification Mathematics of computing → Mathematical analysis, Mathematics of computing → Matchings and factors, Mathematics of computing → Graph algorithms

Keywords and phrases circle squaring, equidecompositions, fractals, point-line incidences

34th International Symposium on Computational Geometry (SoCG 2018).
Editors: Bettina Speckmann and Csaba D. Tóth


Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Sampling Conditions for Conforming Voronoi Meshing by the VoroCrust Algorithm

Ahmed Abdelkader
University of Maryland, College Park MD, USA
akader@cs.umd.edu

Chandrajit L. Bajaj[1]
University of Texas, Austin TX, USA

Mohamed S. Ebeida
Sandia National Laboratories, Albuquerque NM, USA

Ahmed H. Mahmoud
University of California, Davis CA, USA

Scott A. Mitchell
Sandia National Laboratories, Albuquerque NM, USA

John D. Owens
University of California, Davis CA, USA

Ahmad A. Rushdi
University of California, Davis CA, USA

Abstract

We study the problem of decomposing a volume bounded by a smooth surface into a collection of Voronoi cells. Unlike the dual problem of conforming Delaunay meshing, a principled solution to this problem for generic smooth surfaces remained elusive. VoroCrust leverages ideas from α-shapes and the power crust algorithm to produce unweighted Voronoi cells conforming to the surface, yielding the first provably-correct algorithm for this problem. Given an ϵ-sample on the bounding surface, with a weak σ-sparsity condition, we work with the balls of radius δ times the local feature size centered at each sample. The corners of this union of balls are the Voronoi sites, on both sides of the surface. The facets common to cells on opposite sides reconstruct the surface. For appropriate values of ϵ, σ and δ, we prove that the surface reconstruction is isotopic to the bounding surface. With the surface protected, the enclosed volume can be further decomposed into an isotopic volume mesh of fat Voronoi cells by generating a bounded number of sites in its interior. Compared to state-of-the-art methods based on clipping, VoroCrust cells are full Voronoi cells, with convexity and fatness guarantees. Compared to the power crust algorithm, VoroCrust cells are not filtered, are unweighted, and offer greater flexibility in meshing the enclosed volume by either structured grids or random samples.

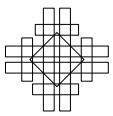
2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases sampling conditions, surface reconstruction, polyhedral meshing, Voronoi

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.1

Related Version The full version is available online [1], http://arxiv.org/abs/1803.06078. In addition, an accompanying multimedia contribution can be found in these proceedings [2], http://dx.doi.org/10.4230/LIPIcs.SoCG.2018.77.

[1] Supported in part by a contract from Sandia, #1439100, and a grant from NIH, R01-GM117594



Funding This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research (ASCR), Applied Mathematics Program.

Acknowledgements We thank Tamal Dey for helpful discussions about surface reconstruction. Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

1 Introduction

Mesh generation is a fundamental problem in computational geometry, geometric modeling, computer graphics, scientific computing and engineering simulations. There has been a growing interest in polyhedral meshes as an alternative to tetrahedral or hex-dominant meshes [48]. Polyhedra are less sensitive to stretching, which enables the representation of complex geometries without excessive refinement. In addition, polyhedral cells have more neighbors even at corners and boundaries, which offers better approximations of gradients and local flow distributions. Even compared to hexahedra, fewer polyhedral cells are needed to achieve a desired accuracy in certain applications. This can be very useful in several numerical methods [18], e.g., finite element [42], finite volume [39], virtual element [17] and Petrov-Galerkin [41]. In particular, the accuracy of a number of important solvers, e.g., the two-point flux approximation for conservation laws [39], greatly benefits from a conforming mesh which is *orthogonal* to its dual as naturally satisfied by Voronoi meshes. Such solvers play a crucial role in hydrology [51], computational fluid dynamics [22] and fracture modeling [20].

VoroCrust is the first provably-correct algorithm for generating a volumetric Voronoi mesh whose boundary conforms to a smooth bounding surface, and with quality guarantees. A conforming volume mesh exhibits two desirable properties *simultaneously*: (1) a decomposition of the enclosed volume, and (2) a reconstruction of the bounding surface.

Conforming Delaunay meshing is well-studied [28], but Voronoi meshing is less mature. A common practical approach to polyhedral meshing is to dualize a tetrahedral mesh and *clip*, i.e., intersect and truncate, each cell by the bounding surface [35,43,47,52,55]. Unfortunately, clipping sacrifices the important properties of convexity and connectedness of cells [35], and may require costly constructive solid geometry operations. Restricting a Voronoi mesh to the surface before *filtering* its dual Delaunay facets is another approach [7,33,56], but filtering requires extra checks complicating its implementation and analysis; see also Figure 4. An intuitive approach is to locally mirror the Voronoi sites on either side of the surface [34,57], but we are not aware of any robust algorithms with approximation guarantees in this category. In contrast to these approaches, VoroCrust is distinguished by its simplicity and robustness at producing true unweighted Voronoi cells, leveraging established libraries, e.g., Voro++ [50], without modification or special cases.

VoroCrust can be viewed as a principled mirroring technique, which shares a number of key features with the power crust algorithm [13]. The power crust literature [7,8,10,12,13] developed a rich theory for surface approximation, namely the ϵ-sampling paradigm. Recall that the power crust algorithm uses an ϵ-sample of unweighted points to place weighted sites, so-called *poles*, near the medial axis of the underlying surface. The surface reconstruction is the collection of facets separating power cells of poles on the inside and outside of the enclosed volume.

Regarding samples and poles as primal-dual constructs, power crust performs a *primal-dual-dual-primal dance*. VoroCrust makes a similar dance where weights are introduced differently; the samples are weighted to define unweighted sites tightly hugging the surface, with the reconstruction arising from their unweighted Voronoi diagram. The key advantage is the freedom to place more sites within the enclosed volume without disrupting the surface reconstruction. This added freedom is essential to the generation of graded meshes; a primary virtue of the proposed algorithm. Another virtue of the algorithm is that all samples appear as vertices in the resulting mesh. While the power crust algorithm does not guarantee that, some variations do so by means of filtering, at the price of the reconstruction no longer being the boundary of power cells [7, 11, 32].

The main construction underlying VoroCrust is a suitable union of balls centered on the bounding surface, as studied in the context of non-uniform approximations [26]. Unions of balls enjoy a wealth of results [15, 24, 37], which enable a variety of algorithms [13, 23, 30].

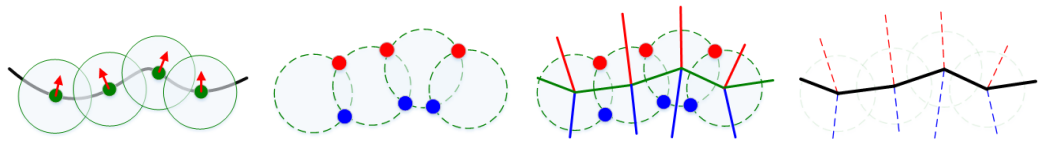
Similar constructions have been proposed for meshing problems in the applied sciences with heuristic extensions to 3D settings; see [40] and the references therein for a recent example. Aichholzer et al. [6] adopt closely related ideas to construct a union of surface balls using power crust poles for sizing estimation. However, their goal was to produce a coarse homeomorphic surface reconstruction. As in [6], the use of balls and α-shapes for surface reconstruction was explored earlier, e.g., ball-pivoting [19, 54], but the connection to Voronoi meshing has been absent. In contrast, VoroCrust aims at a decomposition of the enclosed volume into fat Voronoi cells conforming to an isotopic surface reconstruction with quality guarantees.

In a previous paper [4], we explored the related problem of generating a Voronoi mesh that conforms to restricted classes of piecewise-linear complexes, with more challenging inputs left for future work. The approach adopted in [4] does not use a union of balls and relies instead on similar ideas to those proposed for conforming Delaunay meshing [29, 45, 49].

In this paper, we present a theoretical analysis of an abstract version of the VoroCrust algorithm. This establishes the quality and approximation guarantees of its output for volumes bounded by smooth surfaces. A description of the algorithm we analyze is given next; see Figure 1 for an illustration in 2D, and also our accompanying multimedia contribution [2].

The abstract VoroCrust algorithm

1. Take as input a sample $\mathcal{P}$ on the surface $\mathcal{M}$ bounding the volume $\mathcal{O}$.
2. Define a ball B_i centered at each sample p_i, with a suitable radius r_i, and let $\mathcal{U} = \cup_i B_i$.
3. Initialize the set of sites $\mathcal{S}$ with the corner points of $\partial\mathcal{U}$, $\mathcal{S}^{\uparrow}$ and $\mathcal{S}^{\downarrow}$, on both sides of $\mathcal{M}$.
4. Optionally, generate additional sites $\mathcal{S}^{\downarrow\downarrow}$ in the interior of $\mathcal{O}$, and include $\mathcal{S}^{\downarrow\downarrow}$ into $\mathcal{S}$.
5. Compute the Voronoi diagram $\mathrm{Vor}(\mathcal{S})$ and retain the cells with sites in $\mathcal{S}^{\downarrow} \cup \mathcal{S}^{\downarrow\downarrow}$ as the volume mesh $\hat{\mathcal{O}}$, where the facets between $\mathcal{S}^{\uparrow}$ and $\mathcal{S}^{\downarrow}$ yield a surface approximation $\hat{\mathcal{M}}$.



(a) Surface balls. **(b)** Labeled corners. **(c)** Voronoi cells. **(d)** Reconstruction.

Figure 1 VoroCrust reconstruction, demonstrated on a planar curve.

In this paper, we assume $\mathcal{O}$ is a bounded open subset of $\mathbb{R}^3$, whose boundary $\mathcal{M}$ is a closed, bounded and smooth surface. We further assume that $\mathcal{P}$ is an ϵ-sample, with a weak σ-sparsity condition, and r_i is set to δ times the local feature size at p_i. For appropriate values of ϵ, σ and δ, we prove that $\hat{\mathcal{O}}$ and $\hat{\mathcal{M}}$ are isotopic to $\mathcal{O}$ and $\mathcal{M}$, respectively. We also show that simple techniques for sampling within $\mathcal{O}$, e.g., octree refinement, guarantee an upper bound on the fatness of all cells in $\hat{\mathcal{O}}$, as well as the number of samples.

Ultimately, we seek a conforming Voronoi mesher that can handle realistic inputs possibly containing sharp features, can estimate a sizing function and generate samples, and can guarantee the quality of the output mesh. This is the subject of a forthcoming paper [3] which describes the design and implementation of the complete VoroCrust algorithm.

The rest of the paper is organized as follows. Section 2 introduces the key definitions and notation used throughout the paper. Section 3 describes the placement of Voronoi seeds and basic properties of our construction assuming the union of surface balls satisfies a structural property. Section 4 proves this property holds and establishes the desired approximation guarantees under certain conditions on the input sample. Section 5 considers the generation of interior samples and bounds the fatness of all cells in the output mesh. Section 6 concludes the paper with pointers for future work. A number of proofs is deferred to the full version, available online [1]; see also the accompanying multimedia contribution in these proceedings [2].

2 Preliminaries

Throughout, standard general position assumptions [38] are made implicitly to simplify the presentation. We use $\mathbf{d}(p, q)$ to denote the Euclidean distance between two points $p, q \in \mathbb{R}^3$, and $\mathbb{B}(c, r)$ to denote the Euclidean ball centered at $c \in \mathbb{R}^3$ with radius r. We proceed to introduce the notation and recall the key definitions used throughout, following those in [13, 26, 37].

2.1 Sampling and approximation

We take as input a set of sample points $\mathcal{P} \subset \mathcal{M}$. A local scale or *sizing* is used to vary the sample density. Recall that the *medial axis* [13] of $\mathcal{M}$, denoted by $\mathcal{A}$, is the closure of the set of points in $\mathbb{R}^3$ with more than one closest point on $\mathcal{M}$. Hence, $\mathcal{A}$ has one component inside $\mathcal{O}$ and another outside. Each point of $\mathcal{A}$ is the center of a *medial ball* tangent to $\mathcal{M}$ at multiple points. Likewise, each point on $\mathcal{M}$ has two tangent medial balls, not necessarily of the same size. The *local feature size* at $x \in \mathcal{M}$ is defined as $\mathrm{lfs}(x) = \inf_{a \in \mathcal{A}} \mathbf{d}(x, a)$. The set $\mathcal{P}$ is an *ϵ-sample* [9] if for all $x \in \mathcal{M}$ there exists $p \in \mathcal{P}$ such that $\mathbf{d}(x, p) \leq \epsilon \cdot \mathrm{lfs}(x)$.

We desire an approximation of $\mathcal{O}$ by a Voronoi mesh $\hat{\mathcal{O}}$, where the boundary $\hat{\mathcal{M}}$ of $\hat{\mathcal{O}}$ approximates $\mathcal{M}$. Recall that two topological spaces are *homotopy-equivalent* [26] if they have the same topology type. A stronger notion of topological equivalence is *homeomorphism*, which holds when there exists a continuous bijection with a continuous inverse from $\mathcal{M}$ to $\hat{\mathcal{M}}$. The notion of isotopy captures an even stronger type of equivalence for surfaces *embedded* in Euclidean space. Two surfaces $\mathcal{M}, \hat{\mathcal{M}} \subset \mathbb{R}^3$ are *isotopic* [16, 25] if there is a continuous mapping $F : \mathcal{M} \times [0, 1] \to \mathbb{R}^3$ such that for each $t \in [0, 1]$, $F(\cdot, t)$ is a homeomorphism from $\mathcal{M}$ to $\hat{\mathcal{M}}$, where $F(\cdot, 0)$ is the identity of $\mathcal{M}$ and $F(\mathcal{M}, 1) = \hat{\mathcal{M}}$. To establish that two surfaces are *geometrically close*, the distance between each point on one surface and its closest point on the other surface is required. Such a bound is usually obtained in the course of proving isotopy.

2.2 Diagrams and triangulations

The set of points defining a Voronoi diagram are traditionally referred to as *sites* or *seeds*. When approximating a manifold by a set of sample points of varying density, it is helpful to assign weights to the points reflective of their density. In particular, a point p_i with weight w_i, can be regarded as a ball B_i with center p_i and radius $r_i = \sqrt{w_i}$.

Recall that the *power distance* [37] between two points p_i, p_j with weights w_i, w_j is $\pi(p_i, p_j) = \mathbf{d}(p_i, p_j)^2 - w_i - w_j$. Unless otherwise noted, points are *unweighted*, having weight equal to zero. There is a natural geometric interpretation of the weight: all points q on the boundary of B_i have $\pi(p_i, q) = 0$, inside $\pi(p_i, q) < 0$ and outside $\pi(p_i, q) > 0$. Given a set of weighted points $\mathcal{P}$, this metric gives rise to a natural decomposition of $\mathbb{R}^3$ into the *power cells* $V_i = \{q \in \mathbb{R}^3 \mid \pi(p_i, q) \leq \pi(p_j, q)\ \forall p_j \in \mathcal{P}\}$. The *power diagram* $\mathrm{wVor}(\mathcal{P})$ is the cell complex defined by collection of cells V_i for all $p_i \in \mathcal{P}$.

The nerve [37] of a collection $\mathcal{C}$ of sets is defined as $\mathcal{N}(\mathcal{C}) = \{X \subseteq \mathcal{C} \mid \cap T \neq \emptyset\}$. Observe that $\mathcal{N}(\mathcal{C})$ is an abstract simplicial complex because $X \in \mathcal{N}(\mathcal{C})$ and $Y \subseteq X$ imply $Y \in \mathcal{N}(\mathcal{C})$. With that, we obtain the *weighted Delaunay triangulation*, or *regular triangulation*, as $\mathrm{wDel}(\mathcal{P}) = \mathcal{N}(\mathrm{wVor}(\mathcal{P}))$. Alternatively, $\mathrm{wDel}(\mathcal{P})$ can be defined directly as follows. A subset $T \subset \mathbb{R}^d$, with $d \leq 3$ and $|T| \leq d+1$ defines a d-simplex σ_T. Recall that the *orthocenter* [27] of σ_T, denoted by z_T, is the unique point $q \in \mathbb{R}^d$ such that $\pi(p_i, z_T) = \pi(p_j, z_T)$ for all $p_i, p_j \in T$; the *orthoradius* of σ_T is equal to $\pi(p, z_T)$ for any $p \in T$. The *Delaunay condition* defines $\mathrm{wDel}(\mathcal{P})$ as the set of tetrahedra σ_T with an *empty orthosphere*, meaning $\pi(p_i, z_T) \leq \pi(p_j, z_T)$ for all $p_i \in T$ and $p_j \in \mathcal{P} \setminus T$, where $\mathrm{wDel}(\mathcal{P})$ includes all faces of σ_T.

There is a natural duality between $\mathrm{wDel}(\mathcal{P})$ and $\mathrm{wVor}(\mathcal{P})$. For a tetrahedron σ_T, the definition of z_T immediately implies z_T is a *power vertex* in $\mathrm{wVor}(\mathcal{P})$. Similarly, for each k-face σ_S of $\sigma_T \in \mathrm{wDel}(\mathcal{P})$ with $S \subseteq T$ and $k+1 = |S|$, there exists a dual $(3-k)$-face σ'_S in $\mathrm{wVor}(\mathcal{P})$ realized as $\cap_{p \in S} V_p$. When $\mathcal{P}$ is unweighted, the same definitions yield the standard (unweighted) Voronoi diagram $\mathrm{Vor}(\mathcal{P})$ and its dual Delaunay triangulation $\mathrm{Del}(\mathcal{P})$.

2.3 Unions of balls

Let $\mathcal{B}$ denote the set of balls corresponding to a set of weighted points $\mathcal{P}$ and define the *union of balls* $\mathcal{U}$ as $\cup\mathcal{B}$. It is quite useful to capture the structure of $\mathcal{U}$ using a combinatorial representation like a simplicial complex [36,37]. Let f_i denote $V_i \cap \partial B_i$ and $\mathcal{F}$ the collection of all such f_i. Observing that $V_i \cap B_j \subseteq V_i \cap B_i\ \forall B_i, B_j \in \mathcal{B}$, f_i is equivalently defined as the spherical part of $\partial(V_i \cap B_i)$. Consider also the decomposition of $\mathcal{U}$ by the cells of $\mathrm{wVor}(\mathcal{P})$ into $\mathcal{C}(\mathcal{B}) = \{V_i \cap B_i \mid B_i \in \mathcal{B}\}$. The *weighted α-complex* $\mathcal{W}(\mathcal{P})$ is defined as the *geometric realization* of $\mathcal{N}(\mathcal{C}(\mathcal{B}))$ [37], i.e., $\sigma_T \in \mathcal{W}$ if $\{V_i \cap B_i \mid p_i \in T\} \in \mathcal{N}(\mathcal{C}(\mathcal{B}))$. It is not hard to see that $\mathcal{W}$ is a subcomplex of $\mathrm{wDel}(\mathcal{P})$.

To see why $\mathcal{W}$ is relevant, consider its *underlying space*; we create a collection containing the convex hull of each simplex in $\mathcal{W}$ and define the *weighted α-shape* $\mathcal{J}(\mathcal{P})$ as the union of this collection. It turns out that the simplices $\sigma_T \in \mathcal{W}$ contained in $\partial\mathcal{J}$ are dual to the faces of $\partial\mathcal{U}$ defined as $\cap_{i \in T} f_i$. Every point $q \in \partial\mathcal{U}$ defined by $\cap_{i \in T_q} f_i$, for $T_q \in \mathcal{B}$ and $k+1 = |T_q|$, witnesses the existence of σ_{T_q} in $\mathcal{W}$; the k-simplex σ_{T_q} is said to be *exposed* and $\partial\mathcal{J}$ can be defined directly as the collection of all exposed simplices [36]. In particular, the *corners* of $\partial\mathcal{U}$ correspond to the facets of $\partial\mathcal{J}$. Moreover, $\mathcal{J}$ is homotopy-equivalent to $\mathcal{U}$ [37].

The union of balls defined using an ϵ-sampling guarantees the approximation of the manifold under suitable conditions on the sampling. Following earlier results on uniform sampling [46], an extension to non-uniform sampling establishes sampling conditions for the isotopic approximation of hypersurfaces and medial axis reconstruction [26].

3 Seed placement and surface reconstruction

We determine the location of Voronoi seeds using the union of balls $\mathcal{U}$. The correctness of our reconstruction depends crucially on how sample balls $\mathcal{B}$ overlap. Assuming a certain structural property on $\mathcal{U}$, the surface reconstruction is embedded in the dual shape $\mathcal{J}$.

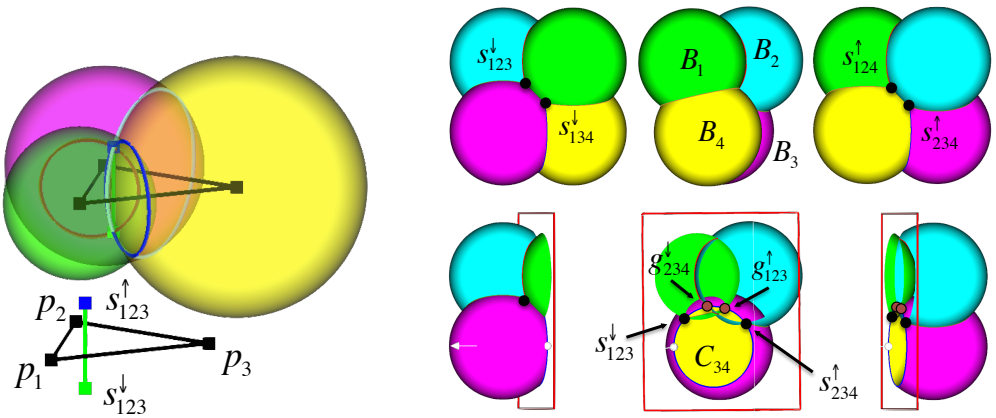
3.1 Seeds and guides

Central to the method and analysis are triplets of sample spheres, i.e., boundaries of sample balls, corresponding to a *guide triangle* in wDel($\mathcal{P}$). The sample spheres associated with the vertices of a guide triangle intersect contributing a pair of *guide points*. The reconstruction consists of Voronoi facets, most of which are guide triangles.

When a triplet of spheres $\partial B_i, \partial B_j, \partial B_k$ intersect at exactly two points, the intersection points are denoted by $g^{\updownarrow}_{ijk} = \{g^{\uparrow}_{ijk}, g^{\downarrow}_{ijk}\}$ and called a pair of *guide points* or *guides*; see Figure 2a. The associated *guide triangle* t_{ijk} is *dual* to $g^{\updownarrow}_{ijk}$. We use arrows to distinguish guides on different sides of the manifold with the *upper* guide $g^{\uparrow}$ lying outside $\mathcal{O}$ and the *lower* guide $g^{\downarrow}$ lying inside. We refer to the edges of guide triangles as *guide edges* $e_{ij} = \overline{p_i p_j}$. A guide edge e_{ij} is associated with a dual *guide circle* $C_{ij} = \partial B_i \cap \partial B_j$, as in Figure 2a.

The Voronoi seeds in $\mathcal{S}^{\uparrow} \cup \mathcal{S}^{\downarrow}$ are chosen as the subset of guide points that lie on $\partial\mathcal{U}$. A guide point g which is not interior to any sample ball is *uncovered* and included as a *seed* s into $\mathcal{S}$; covered guides are not. We denote *uncovered guides* by s and *covered guides* by g, whenever coverage is known and important. If only one guide point in a pair is covered, then we say the guide pair is *half-covered*. If both guides in a pair are covered, they are ignored. Let $\mathcal{S}_i = \mathcal{S} \cap \partial B_i$ denote the seeds on sample sphere ∂B_i.

As each guide triangle t_{ijk} is associated with at least one dual seed s_{ijk}, the seed witnesses its inclusion in $\mathcal{W}$ and t_{ijk} is exposed. Hence, t_{ijk} belongs to $\partial\mathcal{J}$ as well. When such t_{ijk} is dual to a single seeds s_{ijk} it bounds the interior of $\mathcal{J}$, i.e., it is a face of a *regular component* of $\mathcal{J}$; in the simplest and most common case, t_{ijk} is a facet of a tetrahedron as shown in Figure 3b. When t_{ijk} is dual to a pair of seeds $s^{\updownarrow}_{ijk}$, it does not bound the interior of $\mathcal{J}$ and is called a *singular face* of $\partial\mathcal{J}$. All singular faces of $\partial\mathcal{J}$ appear in the reconstructed surface.



(a) Overlapping balls and guide circles. **(b)** Pattern resulting in four half-covered seed pairs.

Figure 2 (a) Guide triangle and its dual seed pair. (b) Cutaway view in the plane of circle C_{34}.

3.2 Disk caps

We describe the structural property required on $\mathcal{U}$ along with the consequences exploited by VoroCrust for surface reconstruction. This is partially motivated by the requirement that all sample points on the surface appear as vertices in the output Voronoi mesh.

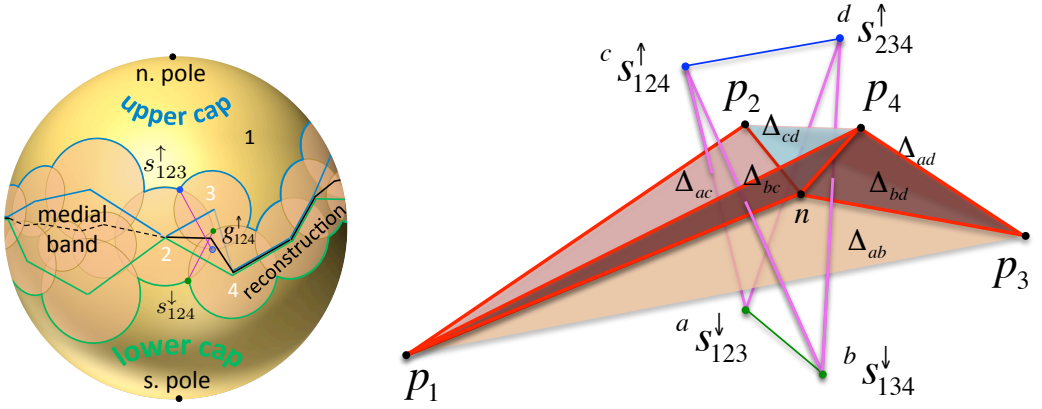
We define the subset of ∂B_i inside other balls as the *medial band* and say it is *covered*. Let the caps $K_i^{\uparrow}$ and $K_i^{\downarrow}$ be the complement of the medial band in the interior and exterior of $\mathcal{O}$, respectively. Letting n_{p_i} be the normal line through p_i perpendicular to $\mathcal{M}$, the two intersection points $n_{p_i} \cap \partial B_i$ are called the *poles* of B_i. See Figure 3a.

We require that $\mathcal{U}$ satisfies the following structural property: each ∂B_i has *disk caps*, meaning the medial band is a *topological annulus* and the two caps contain the poles and are *topological disks*. In other words, each B_i contributes one connected component to each side of $\partial\mathcal{U}$. As shown in Figure 3a, all seeds in $\mathcal{S}_i^{\uparrow}$ and $\mathcal{S}_i^{\downarrow}$ lie on $\partial K_i^{\uparrow}$ and $\partial K_i^{\downarrow}$, respectively, along the arcs where other sample balls intersect ∂B_i. In Section 4, we establish sufficient sampling conditions to ensure $\mathcal{U}$ satisfies this property. In particular, we will show that both poles of each B_i lie on $\partial\mathcal{U}$.

The importance of disk caps is made clear by the following observation. The requirement that all sample points appear as Voronoi vertices in $\hat{\mathcal{M}}$ follows as a corollary.

▶ **Observation 1** (Three upper/lower seeds). *If ∂B_i has* disk caps*, then each of $\partial K_i^{\uparrow}$ and $\partial K_i^{\downarrow}$ has at least three seeds and the seeds on ∂B_i are not all coplanar.*

Proof. Every sphere $S_{j\neq i}$ covers strictly less than one hemisphere of ∂B_i because the poles are uncovered. Hence, each cap is composed of at least three arcs connecting at least three upper seeds $\mathcal{S}_i^{\uparrow} \subset \partial K_i^{\uparrow}$ and three lower seeds $\mathcal{S}_i^{\downarrow} \subset \partial K_i^{\downarrow}$. Further, any hemisphere through the poles contains at least one upper and one lower seed. It follows that the set of seeds $\mathcal{S}_i = \mathcal{S}_i^{\uparrow} \cup \mathcal{S}_i^{\downarrow}$ is not coplanar. ◀



(a) Caps and medial band.

(b) Sliver and half-covered seeds, exaggerated vertical scale.

Figure 3 (a) Decomposing the sample sphere ∂B_1. (b) Uncovered seeds and reconstruction facets. Let $\tau_p \in \mathcal{W}(\mathcal{P}) \subseteq \text{wDel}(\mathcal{P})$ and $\tau_s \in \text{Del}(\mathcal{S})$ denote the tetrahedra connecting the four samples and the four seeds shown, respectively. $s^{\downarrow}_{123}$ and $s^{\downarrow}_{134}$ are the uncovered lower guide seeds, with $g^{\uparrow}_{123}$ and $g^{\uparrow}_{134}$ covered. The uncovered upper guide seeds are $s^{\uparrow}_{124}$ and $s^{\uparrow}_{234}$, with $g^{\downarrow}_{124}$ and $g^{\downarrow}_{234}$ covered. $\triangle_{ac}$ is the Voronoi facet dual to the Delaunay edge between ${}^{a}s^{\downarrow}_{123}$ and ${}^{c}s^{\uparrow}_{124}$, etc. Voronoi facets dual to magenta edges are in the reconstructed surface; those dual to green and blue edges are not. n is the circumcenter of τ_s and appears as a Voronoi vertex in $\text{Vor}(\mathcal{S})$ and a *Steiner vertex* in the surface reconstruction. In general, n is not the orthocenter of the sliver τ_p.

▶ **Corollary 2** (Sample reconstruction). *If ∂B_i has disk caps, then p_i is a vertex in $\hat{\mathcal{M}}$.*

Proof. By Observation 1, the sample is equidistant to at least four seeds which are not all coplanar. It follows that the sample appears as a vertex in the Voronoi diagram and not in the relative interior of a facet or an edge. Being a common vertex to at least one interior and one exterior Voronoi seed, VoroCrust retains this vertex in its output reconstruction. ◀

3.3 Sandwiching the reconstruction in the dual shape of $\mathcal{U}$

Triangulations of smooth surfaces embedded in $\mathbb{R}^3$ can have half-covered guides pairs, with one guide covered by the ball of a fourth sample not in the guide triangle dual to the guide pair. The tetrahedron formed by the three samples of the guide triangle plus the fourth covering sample is a *sliver*, i.e., the four samples lie almost uniformly around the equator of a sphere. In this case we do not reconstruct the guide triangle, and also do not reconstruct some guide edges. We show that the reconstructed surface $\hat{\mathcal{M}}$ lies entirely within the region of space bounded by guide triangles, i.e., the α-shape of $\mathcal{P}$, as stated in the following theorem.

▶ **Theorem 3** (Sandwiching). *If all sample balls have disk caps, then $\hat{\mathcal{M}} \subseteq \mathcal{J}(\mathcal{P})$.*

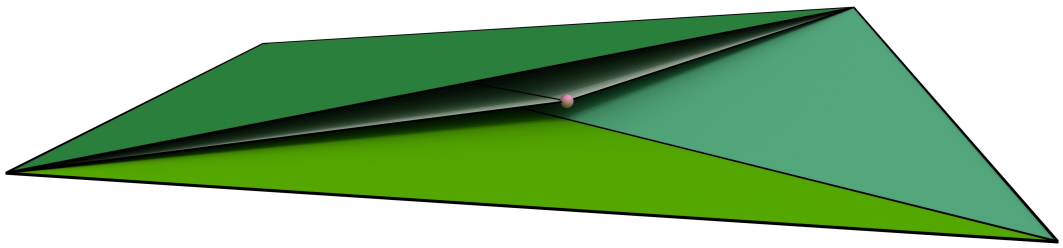


Figure 4 Cutaway view of a sliver tetrahedron $\tau_p \in \mathcal{W}(\mathcal{P}) \subseteq \text{wDel}(\mathcal{P})$, drawn to scale. Half-covered guides give rise to the Steiner vertex (pink), which results in a surface reconstruction using four facets (only two are shown) sandwiched within τ_p. In contrast, filtering wDel($\mathcal{P}$) chooses two of the four facets of τ_p, either the bottom two, or the top two (only one is shown).

The simple case of a single isolated sliver tetrahedron is illustrated in Figures 3b, 4 and 2b. A sliver has a pair of lower guide triangles and a pair of upper guide triangles. For instance, t_{124} and t_{234} are the pair of upper triangles in Figure 3b. In such a tetrahedron, there is an edge between each pair of samples corresponding to a non-empty circle of intersection between sample balls, like the circles in Figure 2a. For this circle, the arcs covered by the two other sample balls of the sliver overlap, so each of these balls contributes exactly one uncovered seed, rather than two. In this way the upper guides for the upper triangles are uncovered, but their lower guides are covered; also only the lower guides of the lower triangles are uncovered. The proof of Theorem 3 follows by analyzing the Voronoi cells of the seed points located on the overlapping sample balls and is deferred to Appendix A [1]. Alternatively, Theorem 3 can be seen as a consequence of Theorem 2 in [15].

4 Sampling conditions and approximation guarantees

We take as input a set of points $\mathcal{P}$ sampled from the bounding surface $\mathcal{M}$ such that $\mathcal{P}$ is an ϵ-sample, with $\epsilon \leq 1/500$. We require that $\mathcal{P}$ satisfies the following sparsity condition: for any two points $p_i, p_j \in P$, $\text{lfs}(p_i) \geq \text{lfs}(p_j) \implies \mathbf{d}(p_i, p_j) \geq \sigma\epsilon\text{lfs}(p_j)$, with $\sigma \geq 3/4$.

Such a sampling $\mathcal{P}$ can be obtained by known algorithms. Given a suitable representation of $\mathcal{M}$, the algorithm in [21] computes a loose ϵ'-sample E which is a $\epsilon'(1+8.5\epsilon')$-sample. More specifically, whenever the algorithm inserts a new sample p into the set E, $\mathbf{d}(p, E) \geq \epsilon' \text{lfs}(p)$. To obtain E as an ϵ-sample, we set $\epsilon'(\epsilon) = (\sqrt{34\epsilon + 1} - 1)/17$. Observing that $3\epsilon/4 \leq \epsilon'(\epsilon)$ for $\epsilon \leq 1/500$, the returned ϵ-sample satisfies our required sparsity condition with $\sigma \geq 3/4$.

We start by adapting Theorem 6.2 and Lemma 6.4 from [26] to the setting just described. For $x \in \mathbb{R}^3 \setminus M$, let $\Gamma(x) = \mathbf{d}(x, \tilde{x})/\text{lfs}(\tilde{x})$, where $\tilde{x}$ is the closest point to x on $\mathcal{M}$.

▶ **Corollary 4.** *For an ϵ-sample $\mathcal{P}$, with $\epsilon \leq 1/20$, the union of balls $\mathcal{U}$ with $\delta = 2\epsilon$ satisfies:*
1. *$\mathcal{M}$ is a deformation retract of $\mathcal{U}$,*
2. *$\partial\mathcal{U}$ contains two connected components, each isotopic to $\mathcal{M}$,*
3. *$\Gamma^{-1}([0, a']) \subset U \subset \Gamma^{-1}([0, b'])$, where $a' = \epsilon - 2\epsilon^2$ and $b' \leq 2.5\epsilon$.*

Proof. Theorem 6.2 from [26] is stated for balls with radii within $[a, b]$ times the lfs. We set $a = b = \delta$ and use $\epsilon \leq 1/20$ to simplify fractions. This yields the above expressions for $a' = (1 - \epsilon)\delta - \epsilon$ and $b' = \delta/(1 - 2\delta)$. The general condition requires $(1 - a')^2 + \big(b' - a' + \delta(1 + 2b' - a')/(1 - \delta)\big)^2 < 1$, as we assume no noise. Plugging in the values of a' and b', we verify that the inequality holds for the chosen range of ϵ. ◀

Furthermore, we require that each ball $B_i \in \mathcal{B}$ contributes one facet to each side of $\partial\mathcal{U}$. Our sampling conditions ensure that both poles are outside any ball $B_j \in \mathcal{B}$.

▶ **Lemma 5** (Disk caps). *All balls in $\mathcal{B}$ have disk caps for $\epsilon \leq 0.066$, $\delta = 2\epsilon$ and $\sigma \geq 3/2$.*

Proof. Fix a sample p_i and let x be one of the poles of B_i and $B_x = \mathbb{B}(c, \text{lfs}(p_i))$ the tangent ball at p_i with $x \in B_x$. Letting p_j be the closest sample to x in $P \setminus \{p_i\}$, we assume the worst case where $\text{lfs}(p_j) \geq \text{lfs}(p_i)$ and p_j lies on ∂B_x. To simplify the calculations, take $\text{lfs}(p_i) = 1$ and let ℓ denote $\mathbf{d}(p_i, p_j)$. As lfs is 1-Lipschitz, we get $\text{lfs}(p_j) \leq 1 + \ell$. By the law of cosines, $\mathbf{d}(p_j, x)^2 = \mathbf{d}(p_i, p_j)^2 + \mathbf{d}(p_i, x)^2 - 2\mathbf{d}(p_i, p_j)\mathbf{d}(p_i, x)\cos(\phi)$, where $\phi = \angle p_j p_i c$. Letting $\theta = \angle p_i c p_j$, observe that $\cos(\phi) = \sin(\theta/2) = \ell/2$. To enforce $x \notin B_j$, we require $\mathbf{d}(p_j, x) > \delta\text{lfs}(p_j)$, which is equivalent to $\ell^2 + \delta^2 - \delta\ell^2 > \delta^2(1 + \ell)^2$. Simplifying, we get $\ell > 2\delta^2/(1 - \delta - \delta^2)$ where sparsity guarantees $\ell > \sigma\epsilon$. Setting $\sigma\epsilon > 2\delta^2/(1 - \delta - \delta^2)$ we obtain $4\sigma\epsilon^2 + (8 + 2\sigma)\epsilon - \sigma < 0$, which requires $\epsilon < 0.066$ when $\sigma \geq 3/4$. ◀

Theorem 4 together with Theorem 5 imply that each ∂B_i is decomposed into a covered region $\partial B_i \cap \cup_{j \neq i} B_j$, the *medial band*, and two uncovered caps $\partial B_i \setminus \cup_{j \neq i} B_j$, each containing one pole. Recalling that seeds arise as pairs of intersection points between the boundaries of such balls, we show that seeds can be classified correctly as either inside or outside $\mathcal{M}$.

▶ **Corollary 6.** *If a seed pair lies on the same side of $\mathcal{M}$, then at least one seed is covered.*

Proof. Fix such a seed pair $\partial B_i \cap \partial B_j \cap \partial B_k$ and recall that $\mathcal{M} \cap \partial B_i$ is contained in the medial band on ∂B_i. Now, assume for contradiction that both seeds are uncovered and lie on the same side of $\mathcal{M}$. It follows that $B_j \cap B_k$ intersects B_i away from its medial band, a contradiction to Theorem 4. ◀

Theorem 4 guarantees that the medial band of B_i is a superset of $\Gamma^{-1}([0, a']) \cap \partial B_i$, which means that all seeds s_{ijk} are at least $a'\text{lfs}(\tilde{s}_{ijk})$ away from $\mathcal{M}$. It will be useful to bound the elevation of such seeds above T_{p_i}, the *tangent plane* to $\mathcal{M}$ at p_i.

▶ **Lemma 7.** *For a seed $s \in \partial B_i$, $\theta_s = \angle s p_i s' \geq 29.34°$ and $\theta_s > \frac{1}{2} - 5\epsilon$, where s' is the projection of s on T_{p_i}, implying $\boldsymbol{d}(s, s') \geq h_s^{\perp} \delta lfs(p_i)$, with $h_s^{\perp} > 0.46$ and $h_s^{\perp} > \frac{1}{2} - 5\epsilon$.*

Proof. Let $\text{lfs}(p_i) = 1$ and $B_s = \mathbb{B}(c, 1)$ be the tangent ball at p_i with $s \notin B_s$; see Figure 5a. Observe that $\mathbf{d}(s, \mathcal{M}) \leq \mathbf{d}(s, x)$, where $x = \overline{sc} \cap \partial B_s$. By the law of cosines, $\mathbf{d}(s, c)^2 = \mathbf{d}(p_i, c)^2 + \mathbf{d}(p_i, s)^2 - 2\mathbf{d}(p_i, c)\mathbf{d}(p_i, s)\cos(\pi/2 + \theta_s) = 1 + \delta^2 + 2\delta\sin(\theta_s)$. We may write[2] $\mathbf{d}(s, c) \leq 1 + \delta^2/2 + \delta\sin(\theta_s)$. It follows that $\mathbf{d}(s, x) \leq \delta^2/2 + \delta\sin(\theta_s)$. As lfs is 1-Lipschitz and $\mathbf{d}(p_i, x) \leq \delta$, we get $1 - \delta \leq \text{lfs}(x) \leq 1 + \delta$. There must exist a sample p_j such that $\mathbf{d}(x, p_j) \leq \epsilon\text{lfs}(x) \leq \epsilon(1 + \delta)$. Similarly, $\text{lfs}(p_j) \geq (1 - \epsilon(1 + \delta))(1 - \delta)$. By the triangle inequality, $\mathbf{d}(s, p_j) \leq \mathbf{d}(s, x) + \mathbf{d}(x, p_j) \leq \delta^2/2 + \delta\sin(\theta_s) + \epsilon(1 + \delta)$. Setting $\mathbf{d}(s, p_j) < \delta(1-\delta)(1-\epsilon(1+\delta))$ implies $\mathbf{d}(s, p_j) < \delta\text{lfs}(p_j)$, which shows that for small values of θ_s, s cannot be a seed and $p_j \neq p_i$. Substituting $\delta = 2\epsilon$, we get $\theta_s \geq \sin^{-1}(2\epsilon^3 - 5\epsilon + 1/2) \geq 29.34°$ and $\theta_s > 1/2 - 5\epsilon$. ◀

We make frequent use of the following bound on the distance between related samples.

▶ **Claim 8.** *If $B_i \cap B_j \neq \emptyset$, then $\boldsymbol{d}(p_i, p_j) \in [\kappa_\epsilon, \kappa\delta] \cdot \textit{lfs}(p_i)$, with $\kappa = 2/(1 - \delta)$ and $\kappa_\epsilon = \sigma\epsilon/(1 + \sigma\epsilon)$.*

Proof. The upper bound comes from $\mathbf{d}(p_i, p_j) \leq r_i + r_j$ and $\text{lfs}(p_j) \leq \text{lfs}(p_i) + \mathbf{d}(p_i, d_j)$ by 1-Lipschitz, and the lower bound from $\text{lfs}(p_i) - \mathbf{d}(p_i, d_j) \leq \text{lfs}(p_j)$ and the sparsity. ◀

Bounding the circumradii is the culprit behind why we need such small values of ϵ.

▶ **Lemma 9.** *The circumradius of a guide triangle t_{ijk} is at most $\varrho_f \cdot \delta\textit{lfs}(p_i)$, where $\varrho_f < 1.38$, and at most $\overline{\varrho}_f \cdot \boldsymbol{d}(p_i, p_j)$ where $\overline{\varrho}_f < 3.68$.*

Proof. Let p_i and p_j be the triangle vertices with the smallest and largest lfs values, respectively. From Claim 8, we get $\mathbf{d}(p_i, p_j) \leq \kappa\delta\text{lfs}(p_i)$. It follows that $\text{lfs}(p_j) \leq (1+\kappa\delta)\text{lfs}(p_i)$. As t_{ijk} is a guide triangle, we know that it has a pair of intersection points $\partial B_i \cap \partial B_j \cap \partial B_k$. Clearly, the seed is no farther than $\delta\text{lfs}(p_j)$ from any vertex of t_{ijk} and the orthoradius of t_{ijk} cannot be bigger than this distance.

Recall that the weight w_i associated with p_i is $\delta^2\text{lfs}(p_i)^2$. We shift the weights of all the vertices of t_{ijk} by the lowest weight w_i, which does not change the orthocenter. With that $w_j - w_i = \delta^2(\text{lfs}(p_j)^2 - \text{lfs}(p_i)^2) \leq \delta^2\text{lfs}(p_i)^2((1 + \kappa\delta)^2 - 1) = \kappa\delta^3\text{lfs}(p_i)^2(\kappa\delta + 2)$. On the other hand, sparsity ensures that the closest vertex in t_{ijk} to p_j is at distance at least $N(p_j) \geq \sigma\epsilon\text{lfs}(p_j) \geq \sigma\epsilon(1 - \kappa\delta)\text{lfs}(p_i)$. Ensuring $\alpha^2 \leq (w_j - w_i)/N(p_i)^2 \leq \kappa\delta^3(2 + \kappa\delta)/(\sigma^2\epsilon^2(1 - \kappa\delta)^2) \leq 1/4$ suffices to bound the circumradius of t_{ijk} by $c_{rad} = 1/\sqrt{1 - 4\alpha^2}$ times its orthoradius, as required by Claim 4 in [27]. Substituting $\delta = 2\epsilon$ and $\sigma \geq 3/4$ we get $\alpha^2 \leq 78.97\epsilon$, which corresponds to $c_{rad} < 1.37$. It follows that the circumradius is at most $c_{rad}\delta\text{lfs}(p_j) \leq c_{rad}(1 + \kappa\delta)\delta\text{lfs}(p_i) < 1.38\delta\text{lfs}(p_i)$.

For the second statement, observe that $\text{lfs}(p_i) \geq (1 - \kappa\delta)\text{lfs}(p_j)$ and the sparsity condition ensures that the shortest edge length is at least $\sigma\epsilon\text{lfs}(p_i) \geq \sigma\epsilon(1 - \kappa\delta)\text{lfs}(p_j)$. It follows that the circumradius is at most $\frac{\delta c_{rad}}{\sigma\epsilon(1-\kappa\delta)} < 3.68$ times the length of any edge of t_{ijk}. ◀

Given the bound on the circumradii, we are able to bound the deviation of normals.

▶ **Lemma 10.** *If t_{ijk} is a guide triangle, then (1) $\angle_a(n_{p_i}, n_{p_j}) \leq \eta_s\delta < 0.47°$, with $\eta_s < 2.03$, and (2) $\angle_a(n_t, n_{p_i}) \leq \eta_t\delta < 1.52°$, with $\eta_t < 6.6$, where n_{p_i} is the line normal to $\mathcal{M}$ at p_i and n_t is the normal to t_{ijk}. In particular, t_{ijk} makes an angle at most $\eta_t\delta$ with T_{p_i}.*

---

[2] Define $f(u, v) = \sqrt{1 + u^2 + 2uv} - (1 + u^2/2 + uv)$ and observe that $f(u, -u/2) = 0$ is the only critical value of $f(u, .)$. As $\partial^2 f/\partial v^2 \leq 0$ for $(u, v) \in \mathbb{R} \times [-1, 1]$, we get that $f(u, v) \leq 0$ in this range.

Proof. Claim 8 implies $\mathbf{d}(p_i, p_j) \leq \kappa\delta\text{lfs}(p_i)$ and (1) follows from the Normal Variation Lemma [14] with $\rho = \kappa\delta < 1/3$ yielding $\angle_a(n_{p_i}, n_{p_j}) \leq \kappa\delta/(1-\kappa\delta)$. Letting R_t denote the circumradius of t, Theorem 9 implies that the $R_t \leq \varrho_f \cdot \delta\text{lfs}(p_i) \leq \text{lfs}(p_i)/\sqrt{2}$ and the Triangle Normal Lemma [31] implies $\angle_a(n_{p^*}, n_t) < 4.57\delta < 1.05°$, where p^* is the vertex of t subtending a maximal angle in t. Hence, $\angle_a(n_{p_i}, n_t) \leq \angle_a(n_{p_i}, n_{p^*}) + \angle_a(n_{p^*}, n_t)$. ◀

Towards establishing homeomorphism, the next lemma on the monotonicity of distance to the nearest seed is critical. First, we show that the nearest seeds to any surface point $x \in \mathcal{M}$ are generated by nearby samples.

▶ **Lemma 11.** *The nearest seed to $x \in \mathcal{M}$ lies on some ∂B_i where $\boldsymbol{d}(x, p_i) \leq 5.03 \cdot \epsilon lfs(x)$. Consequently, $\boldsymbol{d}(x, p_i) \leq 5.08 \cdot \epsilon lfs(p_i)$.*

Proof. In an ϵ-sampling, there exists a p_a such that $\mathbf{d}(x, p_a) \leq \epsilon\text{lfs}(x)$, where $\text{lfs}(p_a) \leq (1+\epsilon)\text{lfs}(x)$. The sampling conditions also guarantee that there exists at least one seed s_a on ∂B_a. By the triangle inequality, we get that $\mathbf{d}(x, s_a) \leq \mathbf{d}(x, p_a) + \mathbf{d}(p_a, s_a) \leq \epsilon\text{lfs}(x) + \delta\text{lfs}(p_a) \leq \epsilon(1 + 2(1+\epsilon))\text{lfs}(x) = \epsilon(2\epsilon+3)\text{lfs}(x)$.

We aim to bound ℓ to ensure $\forall p_i$ s.t. $\mathbf{d}(x, p_i) = \ell \cdot \epsilon\text{lfs}(x)$, the nearest seed to x cannot lie on B_i. Note that in this case, $(1-\ell\epsilon)\text{lfs}(x) \leq \text{lfs}(p_i) \leq (1+\ell\epsilon)\text{lfs}(x)$. Let s_i be any seed on B_i. It follows that $\mathbf{d}(x, s_i) \geq \mathbf{d}(x, p_i) - \mathbf{d}(p_i, s_i) \geq \ell \cdot \epsilon\text{lfs}(x) - 2\epsilon\text{lfs}(p_i) \geq \epsilon\big((1-2\epsilon)\ell - 2\big)\text{lfs}(x)$.

Setting $\epsilon\big((1-2\epsilon)\ell - 2\big)\text{lfs}(x) \geq \epsilon(2\epsilon+3)\text{lfs}(x)$ suffices to ensure $\mathbf{d}(x, s_i) \geq \mathbf{d}(x, s_a)$, and we get $\ell \geq (2\epsilon+5)/(1-2\epsilon)$. Conversely, if the nearest seed to x lies on B_i, it must be the case that $\mathbf{d}(x, p_i) \leq \ell\epsilon\text{lfs}(x)$. We verify that $\ell\epsilon = \epsilon(2\epsilon+5)/(1-2\epsilon) < 1$ for any $\epsilon < 0.13$. It follows that $\mathbf{d}(x, p_j) \leq \ell\epsilon/(1-\ell\epsilon)\text{lfs}(p_i)$. ◀

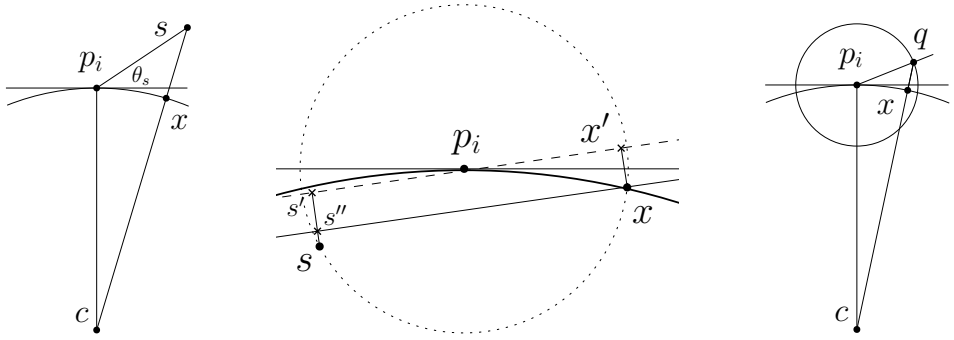
▶ **Lemma 12.** *For any normal segment N_x issued from $x \in \mathcal{M}$, the distance to $\mathcal{S}^{\uparrow}$ is either strictly increasing or strictly decreasing along $\Gamma^{-1}([0, 0.96\epsilon]) \cap N_x$. The same holds for $\mathcal{S}^{\downarrow}$.*

Proof. Let n_x be the outward normal and T_x be the tangent plane to $\mathcal{M}$ at x. By Theorem 11, the nearest seeds to x are generated by nearby samples. Fix one such nearby sample p_i. For all possible locations of a seed $s \in \mathcal{S}^{\uparrow} \cap \partial B_i$, we will show a sufficiently large lower bound on $\langle s - s'', n_x \rangle$, where s'' the projection of s onto T_x.

Take $\text{lfs}(p_i) = 1$ and let $B_s = \mathbb{B}(c, 1)$ be the tangent ball to $\mathcal{M}$ at p_i with $s \in B_s$. Let A be the plane containing $\{p_i, s, x\}$. Assume in the worst case that $A \perp T_{p_i}$ and x is as far as possible from p_i on $\partial B_s \cap T_{p_i}$. By Theorem 11, $\mathbf{d}(p_i, x) \leq 5.08\epsilon$ and it follows that $\theta_x = \angle(n_x, n_{p_i}) \leq 5.08\epsilon/(1-5.08\epsilon) \leq 5.14\epsilon$. This means that T_x is confined within a $(\pi/2 - \theta_x)$-cocone centered at x. Assume in the worst case that n_x is parallel to A and T_x is tilted to minimize $\mathbf{d}(s, s'')$; see Figure 5b.

Let T'_x be a translation of T_x such that $p_i \in T'_x$ and denote by x' and s' the projections of x and s, respectively, onto T'_x. Observe that T'_x makes an angle θ_x with T_{p_i}. From the isosceles triangle $\triangle p_i c x$, we get that $\theta'_x \leq 1/2 \angle p_i c x = \sin^{-1} 5.08\epsilon/2 \leq 2.54\epsilon$. Now, consider $\triangle p_i x x'$ and let $\phi = \angle x p_i x'$. We have that $\phi = \theta_x + \theta'_x \leq 2.54\epsilon + \delta/(1-\delta) \leq 4.55\epsilon$. Hence, $\sin(\phi) \leq 4.55\epsilon$ and $\mathbf{d}(x, x') \leq 5.08\epsilon \sin(\phi) \leq 0.05\epsilon$. On the other hand, we have that $\angle s p_i s' = \psi \geq \theta_s - \theta_x$ and $\mathbf{d}(s, s') \geq \delta \sin\psi$, where $\theta_s \geq 1/2 - 5\epsilon$ by Theorem 7. Simplifying we get $\sin(\psi) \geq 1/2 - 10.08\epsilon$. The proof follows by evaluating $\mathbf{d}(s, s'') = \mathbf{d}(s, s') - \mathbf{d}(x, x')$. ◀

▶ **Theorem 13.** *For every $x \in \mathcal{M}$ with closest point $q \in \hat{\mathcal{M}}$, and for every $q \in \hat{\mathcal{M}}$ with closest point $x \in \mathcal{M}$, we have $\|xq\| < h_t \cdot \epsilon^2 lfs(x)$, where $h_t < 30.52$. For $\epsilon < 1/500$, $h_t \cdot \epsilon^2 < 0.0002$. Moreover, the restriction of the mapping π to $\hat{\mathcal{M}}$ is a homeomorphism and $\hat{\mathcal{M}}$ and $\mathcal{M}$ are ambient isotopic. Consequently, $\hat{\mathcal{O}}$ is ambient isotopic to $\mathcal{O}$ as well.*



(a) Seed elevation θ_s. **(b)** Bounding seed height above T_x. **(c)** Bounding $\mathbf{d}(q, \mathcal{M})$.

Figure 5 Constructions used for (a) Theorem 7, (b) Theorem 12 and (c) Theorem 13.

Proof. Fix a sample $p_i \in \mathcal{P}$ and a surface point $x \in \mathcal{M} \cap B_i$. We consider two cocones centered at x: a p-cocone contains all nearby surface points and a q-cocone contains all guide triangles incident at p_i. By Theorem 3, all reconstruction facets generated by seeds on B_i are sandwiched in the q-cocone.

Theorem 10 readily provides a bound on the q-cocone angle as $\gamma \leq \eta_t \delta$. In addition, since $\mathbf{d}(p_i, x) \leq \delta \mathrm{lfs}(p_i)$, we can bound the p-cocone angle as $\theta \leq 2\sin^{-1}(\delta/2)$ by Lemma 2 in [7]. We utilize a mixed pq-cocone with angle $\omega = \gamma/2 + \theta/2$, obtained by gluing the lower half of the p-cocone with the upper half of the q-cocone.

Let $q \in \hat{\mathcal{M}}$ and consider its closest point $x \in \mathcal{M}$. Again, fix $p_i \in \mathcal{P}$ such that $x \in B_i$; see Figure 5c. By sandwiching, we know that any ray through q intersects at least one guide triangle, in some point y, after passing through x. Let us assume the worst case that y lies on the upper boundary of the pq-cocone. Then, $\mathbf{d}(q, x) \leq \mathbf{d}(y, y') = h = \delta \sin(\omega) \mathrm{lfs}(p_i)$, where y' is the closest point on the lower boundary of the pq-cocone point to q. We also have that, $\mathbf{d}(p_i, x) \leq \cos(\omega)\delta \mathrm{lfs}(p_i) \leq \delta \mathrm{lfs}(p_i)$, and since lfs is 1-Lipschitz, $\mathrm{lfs}(p_i) \leq \mathrm{lfs}(x)/(1-\delta)$. Simplifying, we write $\mathbf{d}(q, x) < \delta\omega/(1-\delta) \cdot \mathrm{lfs}(x) < h_t \epsilon^2 \mathrm{lfs}(x)$.

With $\mathbf{d}(q, x) \leq 0.55\epsilon \mathrm{lfs}(x)$, Theorem 12 shows that the normal line from any $p \in \mathcal{M}$ intersects $\hat{\mathcal{M}}$ exactly once close to the surface. It follows that for every point $x \in \mathcal{M}$ with closest point $q \in \hat{\mathcal{M}}$, we have $\mathbf{d}(x, q) \leq \mathbf{d}(x, q')$ where $q' \in \hat{\mathcal{M}}$ with x its closest point in $\mathcal{M}$. Hence, $\mathbf{d}(x, q) \leq h_t \epsilon^2 \mathrm{lfs}(x)$ as well.

Building upon Theorem 12, as a point moves along the normal line at x, it is either the case that the distance to $\mathcal{S}^{\uparrow}$ is decreasing while the distance to $\mathcal{S}^{\downarrow}$ is increasing or the other way around. It follows that these two distances become equal at exactly one point on the Voronoi facet above or below x separating some seed $s^{\uparrow} \in \mathcal{S}^{\uparrow}$ from another seed $s^{\downarrow} \in \mathcal{S}^{\downarrow}$. Hence, the restriction of the mapping π to $\hat{\mathcal{M}}$ is a homeomorphism.

This shows that $\hat{\mathcal{M}}$ and $\mathcal{M}$ homeomorphic. Recall that Theorem 4(3) implies $\mathcal{U}$ is a *topological thickening* [25] of $\mathcal{M}$. In addition, Theorem 3 guarantees that $\hat{\mathcal{M}}$ is embedded in the interior of $\mathcal{U}$, such that it separates the two surfaces comprising $\partial\mathcal{U}$. These three properties imply $\hat{\mathcal{M}}$ is isotopic to $\mathcal{M}$ in $\mathcal{U}$ by virtue of Theorem 2.1 in [25]. Finally, as $\hat{\mathcal{M}}$ is the boundary of $\hat{\mathcal{O}}$ by definition, it follows that $\hat{\mathcal{O}}$ is isotopic to $\mathcal{O}$ as well. ◀

5 Quality guarantees and output size

We establish a number of quality guarantees on the output mesh. The main result is an upper bound on the *fatness* of all Voronoi cell. See Appendix B for the proofs [1].

Recall that fatness is the outradius to inradius ratio, where the outradius is the radius of the smallest enclosing ball, and the inradius is the radius of the largest enclosed ball. The good quality of guide triangles allows us to bound the inradius of Voronoi cells.

▶ **Lemma 14.** *Consider guide triangle t_{ijk}. (1) Edge length ratios are bounded: $\ell_k/\ell_j \leq \kappa_\ell = \frac{2\delta}{1-\delta}\frac{\sigma\epsilon}{1+\sigma\epsilon}$. (2) Angles are bounded: $\sin(\theta_i) \geq 1/(2\overline{\varrho}_f)$ implying $\theta_i \in (7.8°, 165°)$. (3) Altitudes are bounded: the altitude above e is at least $\alpha_t|e|$, where $\alpha_t = 1/4\overline{\varrho}_f > 0.067$.*

Observe that a guide triangle is contained in the Voronoi cell of its seed, even when one of the guides is covered. Hence, the tetrahedron formed by the triangle together with its seed lies inside the cell, and the cell inradius is at least the tetrahedron inradius.

▶ **Lemma 15.** *For seeds $s_{ijk} \in \mathcal{S}^{\uparrow} \cup \mathcal{S}^{\downarrow}$, the inradius of the Voronoi cell is at least $\varrho_v\delta \cdot lfs(p_i)$ with $\varrho_v = \hat{h}_s/(1 + \frac{3}{2\sigma\overline{\varrho}_f}) > 0.3$ and $\hat{h}_s \geq \frac{1}{2} - (5 + 2\eta_t)\epsilon$.*

To get an upper bound on cell outradii, we must first generate seeds interior to $\mathcal{O}$. We consider a simple algorithm for generating $\mathcal{S}^{\downarrow\downarrow}$ based on a standard octree over $\mathcal{O}$. For sizing, we extend lfs beyond $\mathcal{M}$, using the point-wise maximal 1-Lipschitz extension $\text{lfs}(x) = \inf_{p\in\mathcal{M}}(\text{lfs}(p) + \mathbf{d}(x, p))$ [44]. An octree box $\square$ is refined if the length of its diagonal is greater than $2\delta \cdot \text{lfs}(c)$, where c is the center of $\square$. After refinement terminates, we add an interior seed at the center of each empty box, and do nothing with boxes containing one or more guide seeds. Applying this scheme, we obtain the following.

▶ **Lemma 16.** *The fatness of interior cells is at most $\frac{8\sqrt{3}(1+\delta)}{1-3\delta} < 14.1$.*

▶ **Lemma 17.** *The fatness of boundary cells is at most $\frac{4(1+\delta)}{(1-3\delta)(1-\delta)^2\varrho_v} < 13.65$.*

As the integral of lfs^{-3} is bounded over a single cell, it effectively counts the seeds.

▶ **Lemma 18.** $|\mathcal{S}| \leq 18\sqrt{3}/\pi \cdot \epsilon^{-3} \int_{\mathcal{O}} lfs^{-3}$.

6 Conclusions

We have analyzed an abstract version of the VoroCrust algorithm for volumes bounded by smooth surfaces. We established several guarantees on its output, provided the input samples satisfy certain conditions. In particular, the reconstruction is isotopic to the underlying surface and all 3D Voronoi cells have bounded fatness, i.e., outradius to inradius ratio. The triangular faces of the reconstruction have bounded angles and edge-length ratios, except perhaps in the presence of slivers. In a forthcoming paper [3], we describe the design and implementation of the complete VoroCrust algorithm, which generates conforming Voronoi meshes of realistic models, possibly containing sharp features, and produces samples that follow a natural sizing function and ensure output quality.

For future work, it would be interesting to ensure both guides are uncovered, or both covered. The significance would be that no tetrahedral slivers arise and no Steiner points are introduced. Further, the surface reconstruction would be composed entirely of guide triangles, so it would be easy to show that triangle normals converge to surface normals as sample density increases. Alternatively, where Steiner points are introduced on the surface, it would be helpful to have conditions that guaranteed the triangles containing Steiner points have good quality. In addition, the minimum edge length in a Voronoi cell can be a limiting factor in certain numerical solvers. Post-processing by mesh optimization techniques [5, 53] can help eliminate short Voronoi edges away from the surface. Finally, we expect that the abstract algorithm analyzed in this paper can be extended to higher dimensions.

References

1. A. Abdelkader, C. Bajaj, M. Ebeida, A. Mahmoud, S. Mitchell, J. Owens, and A. Rushdi. Sampling conditions for conforming Voronoi meshing by the VoroCrust algorithm. *CoRR*, arXiv:1803.06078, 2018. URL: `http://arxiv.org/abs/1803.06078`.
2. A. Abdelkader, C. Bajaj, M. Ebeida, A. Mahmoud, S. Mitchell, J. Owens, and A. Rushdi. VoroCrust Illustrated: Theory and Challenges (Multimedia Contribution). In *34th International Symposium on Computational Geometry (SoCG 2018)*, pages 77:1–77:4, 2018. `doi:10.4230/LIPIcs.SoCG.2018.77`.
3. A. Abdelkader, C. Bajaj, M. Ebeida, A. Mahmoud, S. Mitchell, J. Owens, and A. Rushdi. VoroCrust: Voronoi meshing without clipping. *Manuscript*, In preparation.
4. A. Abdelkader, C. Bajaj, M. Ebeida, and S. Mitchell. A Seed Placement Strategy for Conforming Voronoi Meshing. In *Canadian Conference on Computational Geometry*, 2017.
5. A. Abdelkader, A. Mahmoud, A. Rushdi, S. Mitchell, J. Owens, and M. Ebeida. A constrained resampling strategy for mesh improvement. *Computer Graphics Forum*, 36(5):189–201, 2017.
6. O. Aichholzer, F. Aurenhammer, B. Kornberger, S. Plantinga, G. Rote, A. Sturm, and G. Vegter. Recovering structure from r-sampled objects. *Computer Graphics Forum*, 28(5):1349–1360, 2009.
7. N. Amenta and M. Bern. Surface reconstruction by Voronoi filtering. *Discrete & Computational Geometry*, 22(4):481–504, dec 1999.
8. N. Amenta, M. Bern, and D. Eppstein. The crust and the β-skeleton: Combinatorial curve reconstruction. *Graphical models and image processing*, 60(2):125–135, 1998.
9. N. Amenta, M. Bern, and D. Eppstein. Optimal point placement for mesh smoothing. *Journal of Algorithms*, 30(2):302–322, 1999.
10. N. Amenta, M. Bern, and M. Kamvysselis. A new Voronoi-based surface reconstruction algorithm. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, pages 415–421, 1998.
11. N. Amenta, S. Choi, T. Dey, and N. Leekha. A simple algorithm for homeomorphic surface reconstruction. In *16th Annual Symposium on Computational Geometry*, pages 213–222, 2000.
12. N. Amenta, S. Choi, and R.-K. Kolluri. The power crust. In *Proceedings of the Sixth ACM Symp. on Solid Modeling and Applications*, pages 249–266, 2001.
13. N. Amenta, S. Choi, and R.-K. Kolluri. The power crust, unions of balls, and the medial axis transform. *Computational Geometry*, 19(2):127–153, 2001.
14. N. Amenta and T. Dey. Normal variation for adaptive feature size. *CoRR*, abs/1408.0314, 2014.
15. N. Amenta and R.-K. Kolluri. The medial axis of a union of balls. *Computational Geometry*, 20(1):25–37, 2001. Selected papers from the 12th Annual Canadian Conference.
16. N. Amenta, T. Peters, and A. Russell. Computational topology: ambient isotopic approximation of 2-manifolds. *Theoretical Computer Science*, 305(1):3–15, 2003. Topology in Computer Science.
17. L. Beirão da Veiga, F. Brezzi, L.-D. Marini, and A. Russo. The hitchhiker's guide to the virtual element method. *Mathematical Models and Methods in Applied Sciences*, 24(08):1541–1573, 2014.
18. N. Bellomo, F. Brezzi, and G. Manzini. Recent techniques for PDE discretizations on polyhedral meshes. *Mathematical Models and Methods in Applied Sciences*, 24(08):1453–1455, 2014.
19. F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):349–359, Oct 1999.

20 J. Bishop. Simulating the pervasive fracture of materials and structures using randomly close packed Voronoi tessellations. *Computational Mechanics*, 44(4):455–471, sep 2009.

21 J.-D. Boissonnat and S. Oudot. Provably good sampling and meshing of surfaces. *Graphical Models*, 67(5):405–451, 2005. Solid Modeling and Applications.

22 T. Brochu, C. Batty, and R. Bridson. Matching fluid simulation elements to surface geometry and topology. *ACM Trans. Graph.*, 29(4):47:1–47:9, 2010.

23 F. Cazals, T. Dreyfus, S. Sachdeva, and N. Shah. Greedy geometric algorithms for collection of balls, with applications to geometric approximation and molecular coarse-graining. *Computer Graphics Forum*, 33(6):1–17, 2014.

24 F. Cazals, H. Kanhere, and S. Loriot. Computing the volume of a union of balls: A certified algorithm. *ACM Trans. Math. Softw.*, 38(1):3:1–3:20, 2011.

25 F. Chazal and D. Cohen-Steiner. A condition for isotopic approximation. *Graphical Models*, 67(5):390–404, 2005. Solid Modeling and Applications.

26 F. Chazal and A. Lieutier. Smooth manifold reconstruction from noisy and non-uniform approximation with guarantees. *Computational Geometry*, 40(2):156–170, 2008.

27 S.-W. Cheng, T. Dey, H. Edelsbrunner, M. Facello, and S.-H. Teng. Silver exudation. *J. ACM*, 47(5):883–904, 2000.

28 S.-W. Cheng, T. Dey, and J. Shewchuk. *Delaunay Mesh Generation*. CRC Press, 2012.

29 D. Cohen-Steiner, É.-C. de Verdière, and M. Yvinec. Conforming Delaunay triangulations in 3D. In *Proceedings of the Eighteenth Annual Symposium on Computational Geometry*, SCG '02, pages 199–208, 2002.

30 K. Sykes D. Letscher. On the stability of medial axis of a union of disks in the plane. In *28th Canadian Conference on Computational Geometry, CCCG 2016*, pages 29–33, 2016.

31 T. Dey. *Curve and Surface Reconstruction: Algorithms with Mathematical Analysis*. Cambridge University Press, New York, NY, USA, 2006.

32 T. Dey, K. Li, E. Ramos, and R. Wenger. Isotopic reconstruction of surfaces with boundaries. In *Computer Graphics Forum*, volume 28:5, pages 1371–1382, 2009.

33 T. Dey and L. Wang. Voronoi-based feature curves extraction for sampled singular surfaces. *Computers & Graphics*, 37(6):659–668, 2013. Shape Modeling International (SMI) Conference 2013.

34 L. Duan and F. Lafarge. Image partitioning into convex polygons. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3119–3127, June 2015.

35 M. Ebeida and S. Mitchell. Uniform random Voronoi meshes. In *International Meshing Roundtable (IMR)*, pages 258–275, 2011.

36 H. Edelsbrunner. *Weighted alpha shapes*. University of Illinois at Urbana-Champaign, Department of Computer Science, 1992.

37 H. Edelsbrunner. The union of balls and its dual shape. *Discrete & Computational Geometry*, 13(3):415–440, Jun 1995.

38 H. Edelsbrunner and E.-P. Mücke. Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms. *ACM Trans. Graph.*, 9(1):66–104, 1990.

39 R. Eymard, T. Gallouët, and R. Herbin. Finite volume methods. In *Techniques of Scientific Computing (Part 3)*, volume 7 of *Handbook of Numerical Analysis*, pages 713–1018. Elsevier, 2000.

40 Ø. Klemetsdal, R. Berge, K.-A. Lie, H. Nilsen, and O. Møyner. *SPE-182666-MS*, chapter Unstructured Gridding and Consistent Discretizations for Reservoirs with Faults and Complex Wells. Society of Petroleum Engineers, 2017.

41 D. Kuzmin. A guide to numerical methods for transport equations. *University Erlangen-Nuremberg*, 2010.

42 G. Manzini, A. Russo, and N. Sukumar. New perspectives on polygonal and polyhedral finite element methods. *Mathematical Models and Methods in Applied Sciences*, 24(08):1665–1699, 2014.

43 R. Merland, G. Caumon, B. Lévy, and P. Collon-Drouaillet. Voronoi grids conforming to 3D structural features. *Computational Geosciences*, 18(3):373–383, 2014.

44 G. Miller, D. Talmor, and S.-H. Teng. Data generation for geometric algorithms on non-uniform distributions. *International Journal of Computational Geometry and Applications*, 09(06):577–597, 1999.

45 M. Murphy, D. Mount, and C. Gable. A point-placement strategy for conforming Delaunay tetrahedralization. *International Journal of Computational Geometry & Applications*, 11(06):669–682, 2001.

46 P. Niyogi, S. Smale, and S. Weinberger. Finding the homology of submanifolds with high confidence from random samples. *Discrete & Computational Geometry*, 39(1-3):419–441, 2008.

47 A. Okabe, B. Boots, K. Sugihara, and S.-N. Chiu. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, volume 501. John Wiley & Sons, 2009.

48 M. Peric and S. Ferguson. The advantage of polyhedral meshes. *Dynamics - Issue 24*, page 4–5, Spring 2005. The customer magazine of the CD-adapco Group, currently maintained by Siemens at `http://siemens.com/mdx`. The issue is available at `http://mdx2.plm.automation.siemens.com/magazine/dynamics-24` (accessed March 29, 2018).

49 A. Rand and N. Walkington. Collars and intestines: Practical conforming Delaunay refinement. In *Proceedings of the 18th International Meshing Roundtable*, pages 481–497, 2009.

50 C. Rycroft. Voro++: A three-dimensional Voronoi cell library in C++. *Chaos*, 19(4):–, 2009. Software available online at `http://math.lbl.gov/voro++/`.

51 M. Sents and C. Gable. Coupling LaGrit Unstructured Mesh Generation and Model Setup with TOUGH2 Flow and Transport. *Comput. Geosci.*, 108(C):42–49, 2017.

52 H. Si, K. Gärtner, and J. Fuhrmann. Boundary conforming Delaunay mesh generation. *Computational Mathematics and Mathematical Physics*, 50(1):38–53, 2010.

53 D. Sieger, P. Alliez, and M. Botsch. Optimizing Voronoi diagrams for polygonal finite element computations. In *International Meshing Roundtable (IMR)*, pages 335–350. Springer, 2010.

54 P. Stelldinger. Topologically correct surface reconstruction using alpha shapes and relations to ball-pivoting. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4. IEEE, 2008.

55 D.-M. Yan, W. Wang, B. Lévy, and Y. Liu. Efficient computation of clipped Voronoi diagram for mesh generation. *Computer-Aided Design*, 45(4):843–852, 2013.

56 Dong-Ming Yan, Bruno Lévy, Yang Liu, Feng Sun, and Wenping Wang. Isotropic remeshing with fast and exact computation of restricted Voronoi diagram. *Computer Graphics Forum*, 28(5):1445–1454, jul 2009.

57 M. Yip, J. Mohle, and J. Bolander. Automated modeling of three-dimensional structural components using irregular lattices. *Computer-Aided Civil and Infrastructure Engineering*, 20(6):393–407, 2005.

Approximating Maximum Diameter-Bounded Subgraph in Unit Disk Graphs

A. Karim Abu-Affash[1]
Software Engineering Department, Shamoon College of Engineering
Beer-Sheva 84100, Israel
abuaa1@sce.ac.il

Paz Carmi[2]
Department of Computer Science, Ben-Gurion University
Beer-Sheva 84105, Israel
carmip@cs.bgu.ac.il

Anil Maheshwari[3]
School of Computer Science, Carleton University
Ottawa, Canada
anil@scs.carleton.ca

Pat Morin[4]
School of Computer Science, Carleton University
Ottawa, Canada
morin@scs.carleton.ca

Michiel Smid[5]
School of Computer Science, Carleton University
Ottawa, Canada
michiel@scs.carleton.ca

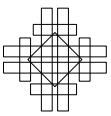
Shakhar Smorodinsky[6]
Department of Mathematics, Ben-Gurion University
Beer-Sheva 84105, Israel
shakhar@math.bgu.ac.il

Abstract

We consider a well studied generalization of the maximum clique problem which is defined as follows. Given a graph G on n vertices and an integer $d \geq 1$, in the *maximum diameter-bounded subgraph* problem (MaxDBS for short), the goal is to find a (vertex) maximum subgraph of G of diameter at most d. For $d = 1$, this problem is equivalent to the maximum clique problem and thus it is NP-hard to approximate it within a factor $n^{1-\epsilon}$, for any $\epsilon > 0$. Moreover, it is known that, for any $d \geq 2$, it is NP-hard to approximate MaxDBS within a factor $n^{1/2-\epsilon}$, for any $\epsilon > 0$.

In this paper we focus on MaxDBS for the class of unit disk graphs. We provide a polynomial-time constant-factor approximation algorithm for the problem. The approximation ratio of our algorithm does not depend on the diameter d. Even though the algorithm itself is simple, its analysis is rather involved. We combine tools from the theory of hypergraphs with bounded VC-dimension, k-quasi planar graphs, fractional Helly theorems and several geometric properties of unit disk graphs.

[1] Work was supported by Grant 2016116 from the United States – Israel Binational Science Foundation.
[2] Work was supported by Grant 2016116 from the United States – Israel Binational Science Foundation.
[3] Work was supported by NSERC
[4] Work was supported by NSERC
[5] Work was supported by NSERC
[6] Work was partially supported by Grant 635/16 from the Israel Science Foundation



34th International Symposium on Computational Geometry (SoCG 2018).
Editors: Bettina Speckmann and Csaba D. Tóth; Article No. 2; pp. 2:1–2:12

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

2012 ACM Subject Classification Theory of computation → Computational geometry, Theory of computation → Numeric approximation algorithms

Keywords and phrases Approximation algorithms, maximum diameter-bounded subgraph, unit disk graphs, fractional Helly theorem, VC-dimension

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.2

Acknowledgements The authors would like to thank the Fields Institute for hosting the workshop in Ottawa and for their financial support.

1 Introduction

Computing a maximum size clique in a graph is one of the fundamental problems in theoretical computer science [7]. It is not only NP-hard but even hard to approximate within a factor of $n^{1-\epsilon}$, for any $\epsilon > 0$, unless $P = NP$ [16]. A clique, equivalently, is a subgraph of diameter 1. A natural, well studied, generalization of the maximum clique problem is the *maximum diameter-bounded subgraph* problem (MaxDBS for short), in which the goal is to compute a maximum (vertex) subgraph of diameter d, for a given $d \geq 2$. A subgraph with diameter $\leq d$ is sometimes referred to in the literature as a *d-club*. MaxDBS is also known to have hardness of approximation of $n^{1/2-\epsilon}$, unless $P = NP$ [5].

In this paper, we study MaxDBS in the class of *unit disk graphs*. A *unit disk graph* is defined as the intersection graph of disks of equal (e.g., unit) diameter in the plane. Unit disk graphs provide a graph-theoretic model for ad hoc wireless networks, where two wireless nodes can communicate if they are within the unit Euclidean distance away from each other.

Many classical NP-Complete problems including chromatic number, independent set and dominating set are still NP-complete even for unit disk graphs [13, 14]. However, the class of unit disk graphs is one of the non-trivial classes of graphs for which the maximum clique problem is in P. Indeed, in a celebrated result, Clark, Colbourn and Johnson [13] provide a beautiful polynomial time algorithm to compute the maximum clique in unit disk graphs. Unfortunately, we do not know how to extend this algorithm to the MaxDBS problem.

Our main contribution in this paper is a polynomial-time algorithm that approximates MaxDBS within a constant factor in unit disk graphs. Moreover, our constant factor approximation ratio does not depend on the diameter d. To the best of our knowledge, it is not known whether this problem is NP-hard. Our algorithm works as follows: Given a unit disk graph $G = (V, E)$ and an integer d, compute for each vertex $v \in V$ the BFS-tree of radius $\frac{d}{2}$ (or $\frac{d+1}{2}$ if d is odd) centered at v, and return the tree of maximum size. Although, the algorithm is very natural and simple, its analysis for unit disk graphs is rather involved. We note that this algorithm might perform very bad for arbitrary graphs. For example, a 2-subdivision of a clique with n vertices is the graph obtained by subdividing each edge of the clique K_n into a path of length 2. It is easily seen that such a graph with $n + \binom{n}{2}$ vertices has diameter 4 but every BFS tree of radius 2 contains at most $2n$ vertices.

1.1 Related work

MaxDBS has been studied extensively in general graphs in the last two decades. Bourjolly et al. [8] showed that MaxDBS is NP-hard. Balasundaram et al. [6] proved that for any d, MaxDBS is NP-hard in graphs of diameter $d + 1$. Asahiro et al. [5] showed that, for any $\epsilon > 0$ and $d \geq 2$, it is NP-hard to approximate MaxDBS within a factor of $n^{1/2-\epsilon}$,

and they gave an $n^{1/2}$-approximation algorithm for the problem. Chang et al. [11] provide an algorithm that finds a maximum subgraph of diameter d in $O(1.62^n \cdot poly(n))$ time. There are more results on solving MaxDBS by using various integer and linear programming formulations [4, 6, 9, 8, 10, 21].

Asahiro et al. [5] studied the MaxDBS in other subclasses of graphs, including chordal graphs, interval graphs, and s-partite graphs. For chordal graphs, they showed that the problem can be solved in polynomial-time for odd d's, and cannot be approximated within factor $n^{1/3-\epsilon}$, for any $\epsilon > 0$ for even d's. For interval graphs, they showed that the problem can be solved in polynomial-time. For s-partite graphs, they showed that the problem cannot be approximated (unless $P = NP$) within a factor of $n^{1/3-\epsilon}$, for any $\epsilon > 0$, when $s = 2$ and $d \geq 3$, and when $s \geq 3$ and $d \geq 2$.

For unit disk graphs, the hardness of MaxDBS is still open, for $d \geq 2$ we are not aware of any previous work. As mentioned already, for $d = 1$, the problem is equivalent to the maximum clique problem and it can be solved in polynomial-time [13].

1.2 Motivation

MaxDBS is a relaxation of the maximum clique problem and is motivated by cluster-detection that arise in wide variety applications. For instance, finding clusters in networks helps in understanding and analyzing the structure of the network. Another well studied notion is that of a *d-clique* [6, 20, 21]. A d-clique of a graph G is a subset S of vertices of G, such that, for every two vertices in S, the shortest distance between them in G is at most d. Clearly, every d-club is a d-clique, but not vice versa, as shown in the example given by Alba [3] in Figure 1.

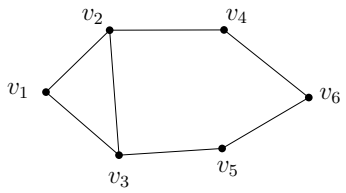


Figure 1 $S = \{v_1, v_2, v_3, v_4, v_5\}$ is a 2-clique but not a 2-club since the graph induced by S has a diameter 3.

2 Preliminaries

Let V be a finite set of points in the plane. For two points $u, v \in V$, let $|uv|$ denote the Euclidean distance between u and v. The unit disk graph on V is the undirected graph $G = (V, E)$, such that $(u, v) \in E$ if and only if $|uv| \leq 1$. The following lemma (though very simple) turns out to be crucial to prove our main result.

▶ **Lemma 1.** *For every two crossing edges (a, b) and (c, d) in G, at least one of the edges (a, c) and (b, d) is in G, and at least one of the edges (a, d) and (b, c) is in G; see Figure 2 for an illustration.*

Proof. To prove the lemma, it suffices to show that $\min\{|ac|, |bd|\} \leq 1$ and $\min\{|ad|, |bc|\} \leq 1$; see Figure 2. Let x be the intersection point of (a, b) and (c, d). By the triangle inequality, $|ac| \leq |ax| + |xc|$ and $|bd| \leq |bx| + |xd|$. Thus, $|ac| + |bd| \leq |ab| + |cd| \leq 2$. Therefore, $\min\{|ac|, |bd|\} \leq 1$. By a similar argument, we prove that $\min\{|ad|, |bc|\} \leq 1$. ◀

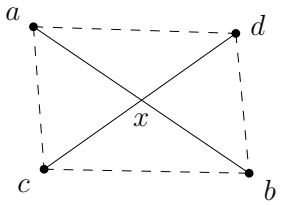


Figure 2 An illustration for the proof of Lemma 1.

2.1 Tool box

A *range space* (or a set system) $(X, \mathcal{R})$ is a pair consisting of a set X of objects (called the *space*) and a family $\mathcal{R}$ of subsets of X (called *ranges*). We say that a subset A of X is *shattered* by $\mathcal{R}$, if for each subset A' of A, there exsist a range $R \in \mathcal{R}$, such that $A \cap R = A'$. The *Vapnik-Chervonenkis* dimension (or VC-dimension for short) of a range space $(X, \mathcal{R})$ is the size of the largest (finite) shattered subset of X; see [15] for examples of range spaces of bounded VC-dimension.

The *dual range space* of $(X, \mathcal{R})$ is a range space $(Y, \mathcal{R}^*)$, where $Y = \{y_R : R \in \mathcal{R}\}$ and, for each $x \in X$, the set $\{y_R : x \in R\}$ is a range in $\mathcal{R}^*$. It is well known [17] that, if the VC-dimension of $(X, \mathcal{R})$ is k, then the VC-dimension of the dual range space $(Y, \mathcal{R}^*)$ is at most 2^k.

A range space $(X, \mathcal{R})$ has fractional Helly number k, if for every $\alpha > 0$, there exists $\beta > 0$, such that if at least $\alpha\binom{|\mathcal{R}|}{k}$ subsets of size k of $\mathcal{R}$ have a non-empty intersection, then there exists an element of X that is contained in at least $\beta \cdot |\mathcal{R}|$ sets of $\mathcal{R}$. In [18], Matoušek proved the following theorem showing that every range space of bounded VC-dimension has a fractional Helly property.

▶ **Theorem 2** ([18]). *Let $(X, \mathcal{R})$ be a range space such that the VC-dimension of the dual range space of $(X, \mathcal{R})$ is at most $k - 1$. Then, $(X, \mathcal{R})$ has a fractional Helly number k.*

A range space $(X, \mathcal{R})$ satisfies the (p, q)-*property* if among every p ranges of $\mathcal{R}$ some q have a non-empty intersection. Matoušek [18] established the following (p, q)-theorem for range spaces of bounded VC-dimension.

▶ **Theorem 3** ([18]). *Let $(X, \mathcal{R})$ be a range space such that the VC-dimension of the dual range space of $(X, \mathcal{R})$ is at most $k - 1$ and let $p \geq k$. Then, there exists a constant t (depending only on p and k), such that if $(X, \mathcal{R})$ satisfies the (p, k)-property, then there exists a subset X' of X of size at most t intersecting all the ranges of $\mathcal{R}$, i.e., $X' \cap R \neq \emptyset$, for every $R \in \mathcal{R}$.*

A (simple) *topological graph* is a graph drawn in the plane, such that its vertices are represented by a set of distinct points and its edges are Jordan arcs connecting the corresponding points, so that (i) each edge does not contain any other vertex as an interior point, (ii) every pair of edges intersect at most once, and (iii) no three edges have a common intersection point. Agarwal et al. [2] showed that any topological graph with n vertices and without k pairwise crossing edges has $O(n \log^{2k-6} n)$ edges. This bound was further improved to $O(n \log^{2k-8} n)$ by Ackerman [1]. Hence, if G is a complete topological graph on n vertices and without k pairwise crossing edges, then $\binom{n}{2} = n(n-1)/2 \leq c' n \log^{2k-8} n$, where c' is the constant in the big 'O', depending only on k. This implies that $k \geq \frac{\log(n-1)-c}{2 \log \log n} + 4$, where c is a constant depending on c'. Therefore, we have the following corollary.

▶ **Corollary 4.** *Any complete topological graph on n vertices contains at least $\frac{\log(n-1)-c}{2\log\log n}+4$ pairwise crossing edges, where c is a constant.*

3 Approximation algorithm

Let $G=(V,E)$ be the unit disk graph of a set of points V in the plane. For two vertices u and v in V, let $d(u,v)$ denote the shortest (hop) distance between u and v in G.[7] Assuming that G is connected, the diameter of G is defined as the maximum (hop) distance between any two vertices in G, i.e., $\max_{u,v\in V} d(u,v)$. A subgraph of G is called *d-club* if its diameter is equal to d. Let G_{opt} denote a maximum d-club of G. In this section, we first present a polynomial-time approximation algorithm that computes a d-club of size at least c times the size of G_{opt}, where c is a constant and d is even. Later, we show how to generalize this algorithm for odd d's.

Set $r=\frac{d}{2}$. For a vertex $u\in V$, let $T_r(u)$ denote the tree of center u and radius r that contains all vertices of distance at most r from u in G. Namely, a vertex v is in $T_r(u)$ if and only if $d(u,v)\leq r$. Given a vertex $u\in V$, $T_r(u)$ can be computed using the breadth first search (BFS) algorithm in $O(|V|+|E|)$ time. Our algorithm computes all the trees of radius r centered at vertices of G and returns a tree T of the maximum size, i.e., the tree that contains the maximum number of vertices. It is clear that T is a d-club of G and can be computed in polynomial time. Let V_{opt} denote the set of vertices of G_{opt} and $n=|V_{opt}|$ denote the size of G_{opt}. In the following, we prove that T contains at least cn vertices, where c is a constant.

Let $\mathcal{T}_r(G_{opt})$ denote the set of all trees of radius r centered at vertices of G_{opt}, and let T_r^* be a tree in $\mathcal{T}_r(G_{opt})$ that contains the maximum number of vertices of G_{opt} among trees of $\mathcal{T}_r(G_{opt})$. First, observe that the size of T is at least as the size of T_r^*, since G_{opt} is a subgraph of G. Therefore, it is sufficient to prove that T_r^* contains at least cn vertices.

Let $(X,\mathcal{R})$ be the range space where $X=V_{opt}$ and for each tree $T_r(u)$ in $\mathcal{T}_r(G_{opt})$, the set $\{v\in V_{opt} : v\in T_r(u)\}$ is a range in $\mathcal{R}$. Thus, in the dual range space $(Y,\mathcal{R}^*)$ of $(X,\mathcal{R})$, we have $Y=\{y_{T_r(u)} : u\in V_{opt}\}$, and, for each point $v\in V_{opt}$, the set $\{y_{T_r(u)} : v\in T_r(u)\}$ is a range in $\mathcal{R}^*$.

▶ **Observation 5.** *$(X,\mathcal{R})$ and $(Y,\mathcal{R}^*)$ are isomorphic.*

Proof. Since $T_r(u)$ contains v if and only if $T_r(v)$ contains u, we have

$$\{y_{T_r(u)} : v\in T_r(u)\}=\{y_{T_r(u)} : u\in T_r(v)\}.$$

Now, if we set $y_{T_r(u)}=u$, then we obtain that $X=Y$ and

$$\{y_{T_r(u)} : v\in T_r(u)\}=\{u\in V_{opt} : v\in T_r(u)\}=\{u\in V_{opt} : u\in T_r(v)\},$$

for each $v\in V_{opt}$. Therefore, for each $v\in V_{opt}$, the set $\{u\in V_{opt} : u\in T_r(v)\}$ is a range in $\mathcal{R}^*$, which implies that $\mathcal{R}^*=\mathcal{R}$. ◀

▶ **Theorem 6.** *T_r^* contains at least cn vertices, where c is a constant.*

Proof. The proof plan is as follows. We show (later in Section 4) that the VC-dimension of the range space $(V_{opt},\mathcal{T}_r(G_{opt}))$ is 4. Thus, by Observation 5, the VC-dimension of the dual

[7] Note that for any $\epsilon>0$, it could hold that the Euclidean distance between u and v is $1+\epsilon$ but they are in different connected components of G and hence $d(u,v)$ is not necessarily bounded.

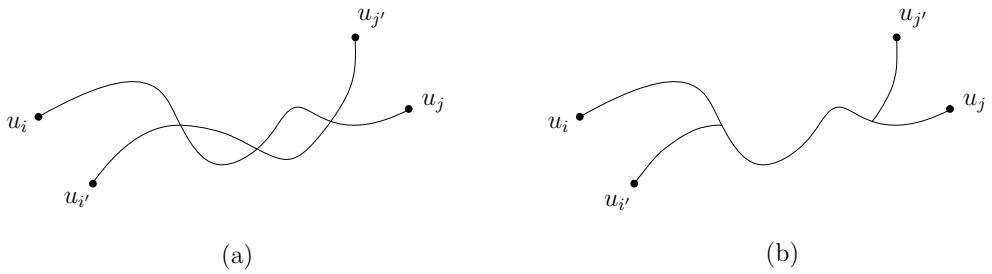


Figure 3 (a) $\delta(u_i, u_j)$ and $\delta(u_{i'}, u_{j'})$ intersect in more than one point, and (b) replacing subpaths of $\delta(u_{i'}, u_{j'})$ by subpaths of $\delta(u_i, u_j)$ between the intersection points.

range space of $(V_{opt}, \mathcal{T}_r(G_{opt}))$ is also 4. Then, we use Corollary 4 to show that there exists a constant $m \geq 5$, such that $(V_{opt}, \mathcal{T}_r(G_{opt}))$ satisfies the $(m, 5)$-property, which means that at least $\binom{n}{m}/\binom{n-5}{m-5} = \binom{n}{5}/\binom{m}{5}$ subsets of 5 trees of $\mathcal{T}_r(G_{opt})$ share a common point. Thus, by Theorem 2, there exists a point that is contained in at least βn trees of $\mathcal{T}_r(G_{opt})$, and therefore, there is a tree in $\mathcal{T}_r(G_{opt})$ that contains at least βn points of V_{opt}, which proves that $c \geq \beta > 0$.

Let m be an integer such that $\frac{\log(m-1)-c'}{2\log\log m} + 4 \geq 6$, where c' is a constant. Let A be a set of m trees of $\mathcal{T}_r(G_{opt})$ and let $C = \{u_1, u_2, \ldots, u_m\}$ be the centers of the trees of A. For two points $u_i, u_j \in C$, let $\delta(u_i, u_j)$ be a shortest path between u_i and u_j in G_{opt}, and let $d(u_i, u_j)$ be the length of $\delta(u_i, u_j)$. For every four distinct points $u_i, u_j, u_{i'}, u_{j'} \in C$, we assume that the intersection of the paths $\delta(u_i, u_j)$ and $\delta(u_{i'}, u_{j'})$ is either empty or a path (otherwise, we replace every subpath of $\delta(u_{i'}, u_{j'})$ between every two consecutive intersection points of the paths by the subpath of $\delta(u_i, u_j)$ between the same points; see Figure 3). Since $d(u_i, u_j) \leq 2r$, there is at least one point $u_{i,j}$ on $\delta(u_i, u_j)$ that is contained in $T_r(u_i)$ and in $T_r(u_j)$.

We now construct a drawing of a complete graph H on the points of C in which the edges are drawn as the Jordan arcs $\delta(u_i, u_j)$. Notice that, H is not necessarily a topological graph. However, we can transform it into a topological graph H', such that, for every four distinct points $u_i, u_j, u_{i'}, u_{j'} \in C$, $\delta(u_i, u_j)$ and $\delta(u_{i'}, u_{j'})$ are crossing in H' if and only if they are crossing in H. This transformation is obtained using standard operations as in [12] and we omit the technical details here. Since H' is a complete topological graph on m vertices, by Corollary 4, H' has at least 6 pairwise crossing edges. Let $P = \{\delta(u_1, u_{1'}), \delta(u_2, u_{2'}), \ldots, \delta(u_6, u_{6'})\}$ be the set of the corresponding 6 pairwise crossing paths in H'.

▶ **Lemma 7.** *Let $\delta(u_i, u_{i'})$ and $\delta(u_j, u_{j'})$ be two crossing paths of P and let x be their intersection point. Assume, w.l.o.g., that x is between $u_{i,i'}$ and u_i, and between $u_{j,j'}$ and u_j; see Figure 4. Then, either $T_r(u_i)$ contains $u_{j,j'}$ or $T_r(u_j)$ contains $u_{i,i'}$.*

Proof. We distinguish between two cases.

Case 1: x is a point of V_{opt}; see Figure 4(a). Assume, w.l.o.g., that $d(u_i, x) \leq d(u_j, x)$. Thus,

$$d(u_i, u_{j,j'}) \leq d(u_i, x) + d(x, u_{j,j'}) \leq d(u_j, x) + d(x, u_{j,j'}) = d(u_j, u_{j,j'}) \leq r.$$

Therefore, $u_{j,j'}$ is of distance at most r from u_i and, hence, is contained in $T_r(u_i)$.

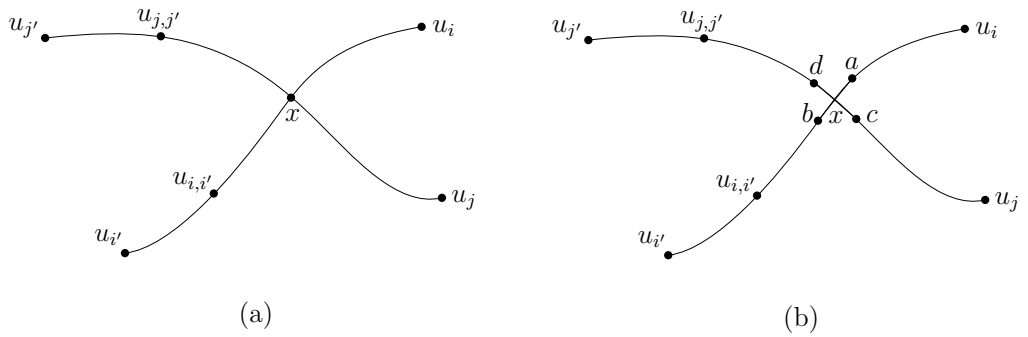


Figure 4 $\delta(u_i, u_{i'})$ and $\delta(u_j, u_{j'})$ intersect at x. (a) x is a point of V_{opt}, and (b) x is an intersection point of the edges (a, b) and (c, d).

Case 2: x is not a point of V_{opt}. Thus, x is an intersection point of two edges (a, b) and (c, d) of G. Assume, w.l.o.g., that a is between x and u_1 and c is between x and u_2; see Figure 4(b).

- If $d(u_i, a) = d(u_j, c)$, then, by Lemma 1, at least one of the edges (a, d) and (b, c) is in G_{opt}.
  - If (a, d) is in G_{opt}, then $d(a, u_{j,j'}) = d(c, u_{j,j'})$, and hence,

$$d(u_i, u_{j,j'}) \leq d(u_i, a) + d(a, u_{j,j'}) = d(u_j, c) + d(c, u_{j,j'}) = d(u_j, u_{j,j'}) \leq r.$$

    Therefore, $u_{j,j'}$ is of distance at most r from u_i and, hence, is contained in $T_r(u_i)$.
  - If (b, c) is in G_{opt}, then $d(a, u_{i,i'}) = d(c, u_{i,i'})$, and hence,

$$d(u_j, u_{i,i'}) \leq d(u_j, c) + d(c, u_{j,j'}) = d(u_i, a) + d(a, u_{i,i'}) = d(u_i, u_{i,i'}) \leq r.$$

    Therefore, $u_{i,i'}$ is of distance at most r from u_j and, hence, is contained in $T_r(u_j)$.
- Otherwise, assume, w.l.o.g., that $d(u_i, a) < d(u_j, c)$. By Lemma 1, at least one of the edges (a, c) and (b, d) is in G_{opt}.
  - If (a, c) is in G_{opt}, then $d(u_i, c) \leq d(u_j, c)$. Hence,

$$d(u_i, u_{j,j'}) \leq d(u_i, c) + d(c, u_{j,j'}) \leq d(u_j, c) + d(c, u_{j,j'}) = d(u_j, u_{j,j'}) \leq r.$$

  - If (b, d) is in G_{opt}, then $d(u_i, d) \leq d(u_j, d)$. Hence,

$$d(u_i, u_{j,j'}) \leq d(u_i, d) + d(d, u_{j,j'}) \leq d(u_j, d) + d(d, u_{j,j'}) = d(u_j, u_{j,j'}) \leq r.$$

  In both cases, $u_{j,j'}$ is of distance at most r from u_i and, hence, is contained in $T_r(u_i)$. ◀

▶ **Lemma 8.** *$(V_{opt}, \mathcal{T}_r(G_{opt}))$ satisfies the $(m, 5)$-property.*

Proof. By Lemma 7, for every two paths $\delta(u_i, u_{i'})$ and $\delta(u_j, u_{j'})$ in P, either at least one of the trees $T_r(u_i)$ and $T_r(u_{i'})$ contains $u_{j,j'}$ or at least one of the trees $T_r(u_j)$ and $T_r(u_{j'})$ contains $u_{i,i'}$. We construct a directed graph on the vertices $\{u_1, u_2, \ldots, u_6\}$, such that there is a directed edge from u_i to u_j if and only if at least one of the trees $T_r(u_i)$ and $T_r(u_{i'})$ contains $u_{j,j'}$. Since we have 6 pairwise crossing paths, there are at least 15 edges in this graph, which means that there is a vertex u_l in this graph, $1 \leq l \leq 6$, of in-degree at least 3. Hence, there is a point $u_{l,l'}$ that is covered by at least 3 other trees, in addition to the trees $T_r(u_l)$ and $T_r(u_{l'})$. Thus, $u_{l,l'}$ is contained in at least 5 trees of A. Therefore, $(V_{opt}, \mathcal{T}_r(G_{opt}))$ satisfies the $(m, 5)$-property. ◀

Since $(V_{opt}, \mathcal{T}_r(G_{opt}))$ satisfies the $(m, 5)$-property, out of every m trees of $\mathcal{T}_r(G_{opt})$ there are 5 trees that share a common point. Thus, there are at least $\binom{n}{m}/\binom{n-5}{m-5} = \binom{n}{5}/\binom{m}{5}$ sets containing 5 trees that share a common point. Moreover, in Theorem 12 (Section 4), we show that the VC-diminsion of the range space $(V_{opt}, \mathcal{T}_r(G_{opt}))$ is 4. Thus, by Observation 5, the VC-dimension of the dual range space of $(V, \mathcal{T}_r(G))$ is also 4 and therefore, by Theorem 2, the fractional Helly number of $(V_{opt}, \mathcal{T}_r(G_{opt}))$ is 5. Now, by setting $\alpha = 1/\binom{m}{5}$ (in the fractional Helly theorem), we have a point $u \in V_{opt}$ that is contained in at least βn trees of $\mathcal{T}_r(G_{opt})$, which means that $T_r(u)$ contains at least βn points of V_{opt}, where $\beta > 0$. Therefore, $c \geq \beta > 0$, which completes the proof of the theorem. ◀

▶ **Corollary 9.** *For any even $d \geq 2$, every d-club in any unit disk graph can be covered by a constant-number of trees of radius $\frac{d}{2}$.*

Proof. To prove the corollary, we show that there exists a constant ρ, such that G_{opt} can be covered by at most ρ trees of $\mathcal{T}_r(G_{opt})$. By Theorem 12, the VC-dimension of the dual range space of $(V_{opt}, \mathcal{T}_r(G_{opt}))$ is 4, and, by Lemma 8, $(V_{opt}, \mathcal{T}_r(G_{opt}))$ satisfies the $(m, 5)$-property. Therefore, by Theorem 3, there exists a set of at most t trees of $\mathcal{T}_r(G_{opt})$ that cover all vertices of V_{opt}, which proves that $\rho \leq t$. ◀

Upper bound on c

We show, in Figure 5, a unit disk graph G on n vertices for which the tree computed by our algorithm does not contain more than $\frac{n}{3}$. G contains $n = 16r$ points and its diameter is $d = 2r$. Each tree of radius r in G covers at most $6r$ points. This proves that $c \leq \frac{3}{8}$. To show that $c \leq \frac{1}{3}$, we locate six cliques of size $\frac{n-16r}{6}$ on the points a, b, c, a', b', and c'. Now, each tree of radiuns r can cover at most 2 cliques. Therefore, for sufficiently large n, we have $c \leq \frac{1}{3}$.

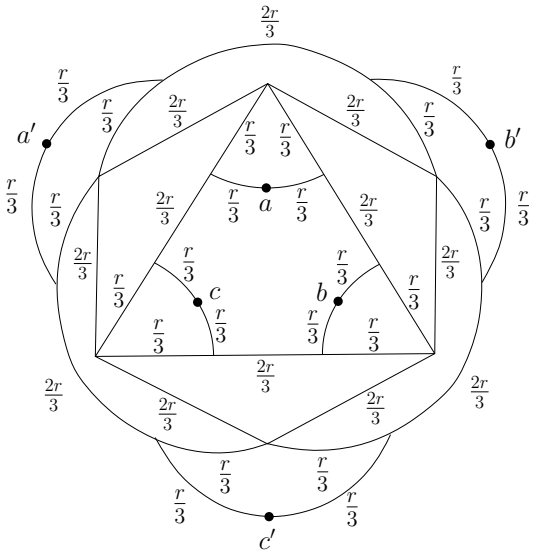


Figure 5 G contains $16r$ points and its diameter is $d = 2r$ Each tree of radius r covers at most $6r$ points.

Generalization for odd d

In this section, we extend our algorithm to approximate MaxDBS for odd d's. Given a unit disk graph G of a set V of points in the plane and an odd integer $d \geq 3$, let G_d be a maximum d-club and let G_{d+1} be a maximum $(d+1)$-club of G. Let n_d and n_{d+1} be the sizes of G_d and G_{d+1}, respectively, and observe that $n_{d+1} \geq n_d$. We set $r = \frac{d+1}{2}$ and we use our algorithm to compute a tree $T_r(u)$ of size at least $cn_{d+1} \geq cn_d$. Notice that $T_r(u)$ is a $(d+1)$-club but may not be a d-club. In the following lemma, we show that there is a subtree of $T_r(u)$ of diameter $d-1$ that contains at least $1/12$ of the vertices of $T_r(u)$.

▶ **Lemma 10.** *The vertices of tree $T_r(u)$ can be covered by at most 12 trees of radius $r-1$.*

Proof. Let $V_r(u)$ be the set of vertices of $T_r(u)$ and let $D_2(u) = \{v \in V_r(u) : d(u,v) = 2\}$, i.e., the set of all vertices of $V_r(u)$ of distance two from u. Let I be a maximal independent set of $D_2(u)$. By the packing argument in unit disk graphs, we have $|I| \leq 12$. Let v be a vertex in $V_r(u)$, and let $\delta(u,v)$ be a shortest path between u and v in $T_r(u)$. Since $d(u,v) \leq r$, there is at least one vertex $u' \in D_2(u)$, such that every vertex in $\delta(u,v)$ is of distance at most $r-2$ from u'. Hence, there is at least one vertex $x \in I$, such that every vertex in $\delta(u,v)$ is of distance at most $r-1$ from x. Thus, every vertex in $V_r(u)$ is contained in $T_{r-1}(x)$, for some $x \in I$, and therefore, $V_r(u)$ is covered by $\bigcup_{x \in I} T_{r-1}(x)$. ◀

By Lemma 10, we can find a tree $T^*_{r-1}(x)$ that contains at least $1/12$ of the vertices of $T_r(u)$. Since $r = \frac{d+1}{2}$, the diameter of $T^*_{r-1}(x)$ is at most $2(r-1) = d-1$. Therefore, $T^*_{r-1}(x)$ is a d-club of G and its size is at least $\frac{c}{12} n_d$.

The following theorem summarizes the result of this section.

▶ **Theorem 11.** *Given a unit disk graph G in the plane and an integer $d \geq 2$, one can find in polynomial time a d-club of G of size at least $\frac{c}{12}$ the size of a maximum d-club of G, where $0 < c \leq \frac{1}{3}$.*

4 The VC-Dimension of $(V_{opt}, \mathcal{T}_r(G_{opt}))$

In this section, we prove the following theorem.

▶ **Theorem 12.** *The range space $(V_{opt}, \mathcal{T}_r(G_{opt}))$ has VC-dimension 4.*

Proof. We first prove that the VC-dimension of $\mathcal{T}_r(G_{opt})$ is at most 4. For the sake of contradiction, suppose that there exist a set of points P and a subset $S = \{u_1, u_2, u_3, u_4, u_5\}$ of V_{opt}, such that S is shattered by $\mathcal{T}_r(G_{opt})$. Thus, for each $1 \leq i < j \leq 5$, there is a tree $T_r(c_{i,j})$ in $\mathcal{T}_r(G_{opt})$, such that $T_r(c_{i,j}) \cap S = \{u_i, u_j\}$. Let $P_{i,j}$ be the path between u_i and u_j in $T_r(c_{i,j})$. Note that $P_{i,j} \cap S = \{u_i, u_j\}$. Moreover, since S contains five points, by planarity constraints, at least two paths $P_{i,j}$ and $P_{k,l}$, for distinct two pairs (i,j) and (k,l), intersect. Assume, w.l.o.g., that $P_{1,3}$ and $P_{2,4}$ intersect and let x be their intersection point. Assume also that x is between u_1 and $c_{1,3}$, and between u_2 and $c_{2,4}$; see Figure 6.

▶ **Lemma 13.** *Either u_1 is contained in $T_r(c_{2,4})$ or u_2 is contained in $T_r(c_{1,3})$.*

Proof. The proof is similar to the proof of Lemma 7. We distinguish between two cases.

Case 1: x is a point of V_{opt}; see Figure 6(a). Assume, w.l.o.g., that $d(x,u_1) \leq d(x,u_2)$. Thus,

$$d(c_{2,4}, u_1) \leq d(c_{2,4}, x) + d(x, u_1) \leq d(c_{2,4}, x) + d(x, u_2) = d(c_{2,4}, u_2) \leq r.$$

Figure 6 $P_{1,3}$ and $P_{2,4}$ intersect at x. (a) x is a point of V_{opt}, and (b) x is an intersection point of the edges (a, b) and (c, d).

Therefore, u_1 is of distance at most r from $c_{2,4}$ and, hence, is contained in $T_r(c_{2,4})$.

Case 2: x is not a point of V_{opt}. Thus, x is an intersection point of two edges (a, b) and (c, d) of G. Assume, w.l.o.g., that a is between x and u_1 and c is between x and u_2; see Figure 6(b).

- If $d(a, u_1) = d(c, u_2)$, then, by Lemma 1, at least one of the edges (a, d) and (b, c) is in G_{opt}. If (a, d) is in G_{opt}, then $d(c_{2,4}, a) = d(c_{2,4}, c)$, and hence,

$$d(c_{2,4}, u_1) \leq d(c_{2,4}, a) + d(a, u_1) = d(c_{2,4}, c) + d(c, u_2) = d(c_{2,4}, u_2) \leq r.$$

  Therefore, u_1 is of distance at most r from $c_{2,4}$ and, hence, is contained in $T_r(c_{2,4})$. If (b, c) is in G_{opt}, then $d(c_{1,3}, a) = d(c_{1,3}, c)$, and hence,

$$d(c_{1,3}, u_2) \leq d(c_{1,3}, c) + d(c, u_2) = d(c_{1,3}, a) + d(a, u_1) = d(c_{1,3}, u_1) \leq r.$$

  Therefore, u_2 is of distance at most r from $c_{1,3}$ and, hence, is contained in $T_r(c_{1,3})$.
- Otherwise, assume, w.l.o.g., that $d(a, u_1) < d(c, u_2)$. By Lemma 1, at least one of the edges (a, c) and (b, d) is in G_{opt}. If (b, d) is in G_{opt}, then $d(c_{2,4}, b) \leq d(c_{2,4}, c)$ and $d(b, u_1) \leq d(c, u_2)$. Hence,

$$d(c_{2,4}, u_1) \leq d(c_{2,4}, b) + d(b, u_1) \leq d(c_{2,4}, c) + d(c, u_2) = d(c_{2,4}, u_2) \leq r.$$

  If (a, c) is in G_{opt}, then $d(c, u_1) \leq d(c, u_2)$. Thus,

$$d(c_{2,4}, u_1) \leq d(c_{2,4}, c) + d(c, u_1) \leq d(c_{2,4}, c) + d(c, u_2) = d(c_{2,4}, u_2) \leq r.$$

In both cases, u_1 is of distance at most r from $c_{2,4}$ and, hence, is contained in $T_r(c_{2,4})$. ◀

Since $T_r(c_{2,4}) \cap S = \{u_2, u_4\}$ and $T_r(c_{1,3}) \cap S = \{u_1, u_3\}$, we have a contradiction. Therefore, the VC-dimension of $(V_{opt}, \mathcal{T}_r(G_{opt}))$ is at most 4.

To prove that the VC-dimension of $(V_{opt}, \mathcal{T}_r(G_{opt}))$ is at least 4, we show in Figure 7 a unit disk graph on a set of points V of diameter $2r$ and a subset $S = \{a, b, c, d\}$ of V, such that S can be shattered by $\mathcal{T}_r(G_{opt})$. The distance between every two points of S is r. For each subset $S' \subset S$, $S \cap T_r(v_{S'}) = S'$, and $S \cap T_r(a) = S$. ◀

5 Concluding remarks

In this paper, we consider the problem of computing a maximum subgraph of diameter d. We present the first constant-factor approximation algorithm for the problem in unit disk graphs, for any $d \geq 2$.

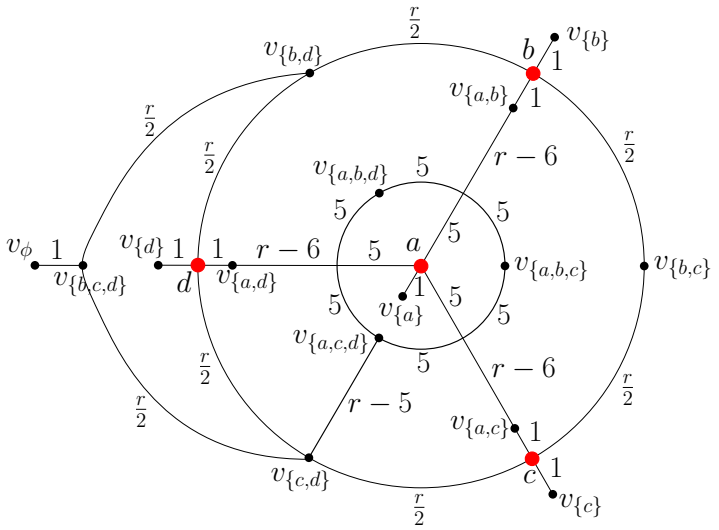


Figure 7 Shattering the points a, b, c, and d. $S \cap T_r(v_{S'}) = S'$, for each $S' \subset S$, and $S \cap T_r(a) = S$.

Our algorithm is simple and efficient, however, its analysis is not trivial and based on tools from the theory of hypergraphs with bounded VC-dimension, k-quasi planar graphs, fractional Helly theorems and several geometric properties of unit disk graphs. Unfortunately, the constant obtained is rather large. On the other hand, the most important feature of our algorithm is that its approximation factor is a constant independent of the diameter d. It is very easy to obtain an $O(d^2)$ approximation factor. Indeed, by a packing argument, any graph of diameter d can be covered by $O(d^2)$ cliques and as mentioned already the max-clique problem is in P.

Moreover, our algorithm works also for an abstract input of the unit disk graph without the geometric representation. It remains an open problem to determine whether MaxDBS for unit disk graphs is in P for $d \geq 2$.

Another interesting fact to note is that the shortest path metric on unit disk graphs does not have the so-called *constant doubling dimension*. It is easily seen that our algorithm has a constant factor approximation for graph families with constant doubling dimension.

Recall that a d-clique of a graph G is a set S of vertices of G, such that, for every two vertices in S, the shortest distance between them in G is at most d. Finding the maximum d-clique problem is closely related to MaxDBS. Unfortunately, our algorithm can not be directly extended to the maximum d-clique problem. Except for the $\frac{1}{2}$-approximation algorithm of Pattillo et al. [19], for $d = 2$, there is no related work discussing the maximum d-clique problem in unit disk graphs. Hence, approximating the maximum d-clique problem in unit disk graphs is also an interesting open problem.

References

1 E. Ackerman. On the maximum number of edges in topological graphs with no four pairwise crossing edges. *Discrete Comput. Geom.*, 41:365–375, 2009.

2 P. K. Agarwal, B. Aronov, J. Pach, and M. Sharir. Quasi-planar graphs have linear number of edges. *Combinatorica*, 17:1–9, 1997.

3 R. D. Alba. A graph-theoretic definition of a sociometric clique. *J. Math. Sociol.*, 3:113–126, 1973.

4 M. T. Almeida and F. D. Carvalho. Integer models and upper bounds for the 3-club problem. *Networks*, 60:155–166, 2012.

5 Y. Asahiro, E. Miyano, and K. Samizo. Approximating maximum diameter-bounded subgraphs. In *LATIN, LNCS 6034*, pages 615–626, 2010.

6 B. Balasundaram, S. Butenko, and Trukhanov S. Novel approaches for analyzing biological networks. *J. Combin. Optim.*, 10:23–39, 2005.

7 I. M. Bomze, M. Budinich, P. M. Pardalos, and M. Pelillo. The maximum clique problem. *Handbook of Combinatorial Optimization*, pages 1–74, Kluwer Academic Publishers, 1999.

8 J.-M. Bourjolly, G. Laporte, and G. Pesant. An exact algorithm for the maximum k-club problem in an undirected graph. *European J. Oper. Res.*, 138:21–28, 2002.

9 A. Buchanan and H. Salemi. Parsimonious formulations for low-diameter clusters. http://www.optimization-online.org/DB_HTML/2017/09/6196.html, 2017.

10 F. D. Carvalho and M. T. Almeida. Upper bounds and heuristics for the 2-club problem. *European J. Oper. Res.*, 210:489–494, 2011.

11 M.-S. Chang, L.-J. Hung, C.-R. Lin, and P.-C. Su. Finding large k-clubs in undirected graphs. *Computing*, 95:739–758, 2013.

12 V. Chepoi, B. Estellon, and Y. Vaxès. Covering planar graphs with a fixed number of balls. *Discrete Comput. Geom.*, 37:237–244, 2007.

13 B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit disk graphs. *Discrete Math.*, 86:165–177, 1990.

14 A. Gräf, M. Stumpf, and G. Weißenfels. On coloring unit disk graphs. *Algorithmica*, 20(3):277–293, 1998.

15 S. Har-Peled. *Geometric Approximation Algorithms*. American Mathematical Society, Boston, USA, 2011.

16 J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. In *FOCS*, pages 627–636, 1996.

17 J. Matoušek. *Lectures on Discrete Geometry*. Springer, New York, USA, 2002.

18 J. Matoušek. Bounded VC-dimension implies a fractional Helly theorem. *Discrete Comput. Geom.*, 31:251–255, 2004.

19 J. Pattillo, Y. Wang, and S. Butenko. Approximating 2-cliques in unit disk graphs. *Discrete Appl. Math.*, 166:178–187, 2014.

20 J. Pattillo, N. Youssef, and S. Butenko. On clique relaxation models in network analysis. *European J. Oper. Res.*, 226:9–18, 2013.

21 A. Veremyev and V. Boginski. Identifying large robust network clusters via new compact formulations of maximum k-club problems. *European J. Oper. Res.*, 218:316–326, 2012.

Vietoris–Rips and Čech Complexes of Metric Gluings

Michał Adamaszek
MOSEK ApS
Copenhagen, Denmark
aszek@mimuw.edu.pl
https://orcid.org/0000-0003-3551-192X

Henry Adams
Department of Mathematics, Colorado State University
Fort Collins, CO, USA
adams@math.colostate.edu
https://orcid.org/0000-0003-0914-6316

Ellen Gasparovic
Department of Mathematics, Union College
Schenectady, NY, USA
gasparoe@union.edu
https://orcid.org/0000-0003-3775-9785

Maria Gommel
Department of Mathematics, University of Iowa
Iowa City, IA, USA
maria-gommel@uiowa.edu
https://orcid.org/0000-0003-2714-9326

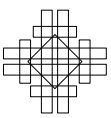
Emilie Purvine
Computing and Analytics Division, Pacific Northwest National Laboratory
Seattle, WA, USA
emilie.purvine@pnnl.gov
https://orcid.org/0000-0003-2069-5594

Radmila Sazdanovic[1]
Department of Mathematics, North Carolina State University
Raleigh, NC, USA
rsazdanovic@math.ncsu.edu
https://orcid.org/0000-0003-1321-1651

Bei Wang[2]
School of Computing, University of Utah
Salt Lake City, UT, USA
beiwang@sci.utah.edu
https://orcid.org/0000-0002-9240-0700

Yusu Wang[3]
Department of Computer Science, The Ohio State University
Columbus, OH, USA
yusu@cse.ohio-state.edu
https://orcid.org/0000-0001-7950-4348

[1] Simons Collaboration Grant 318086
[2] NSF IIS-1513616 and NSF ABI-1661375
[3] NSF CCF-1526513, CCF-1618247, and CCF-1740761



34th International Symposium on Computational Geometry (SoCG 2018).
Editors: Bettina Speckmann and Csaba D. Tóth; Article No. 3; pp. 3:1–3:15

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Lori Ziegelmeier
Department of Mathematics, Statistics, and Computer Science, Macalester College
Saint Paul, MN, USA
lziegel1@macalester.edu
https://orcid.org/0000-0002-1544-4937

Abstract

We study Vietoris–Rips and Čech complexes of metric wedge sums and metric gluings. We show that the Vietoris–Rips (resp. Čech) complex of a wedge sum, equipped with a natural metric, is homotopy equivalent to the wedge sum of the Vietoris–Rips (resp. Čech) complexes. We also provide generalizations for certain metric gluings, i.e. when two metric spaces are glued together along a common isometric subset. As our main example, we deduce the homotopy type of the Vietoris–Rips complex of two metric graphs glued together along a sufficiently short path. As a result, we can describe the persistent homology, in all homological dimensions, of the Vietoris–Rips complexes of a wide class of metric graphs.

2012 ACM Subject Classification Theory of computation → Computational geometry, Mathematics of computing → Topology

Keywords and phrases Vietoris–Rips and Čech complexes, metric space gluings and wedge sums, metric graphs, persistent homology

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.3

Related Version A full version is available at `https://arxiv.org/abs/1712.06224`.

Funding We are grateful for the Women in Computational Topology (WinCompTop) workshop for initiating our research collaboration. In particular, participant travel support was made possible through the grant NSF-DMS-1619908. Our group was also supported by the American Institute of Mathematics (AIM) Structured Quartet Research Ensembles (SQuaRE) program.

1 Introduction

When equipped with a notion of similarity or distance, data can be thought of as living in a metric space. Our goal is to characterize the homotopy types of geometric thickenings of a wide class of metric spaces. In particular, we consider metric spaces formed by gluing smaller metric spaces together along certain nice intersections; our results then characterize the persistent homology of these spaces. Persistent homology is a central tool in topological data analysis that captures complex interactions within a system at multiple scales [13, 19].

The geometric complexes of interest are Vietoris–Rips and Čech complexes, which build a simplicial complex on top of a metric space according to the choice of a scale parameter r. We first study Vietoris–Rips and Čech complexes of metric wedge sums: given two metric spaces X and Y with specified basepoints, the metric wedge sum $X \vee Y$ is obtained by gluing X and Y together at the specified points. We show that the Vietoris–Rips (resp. Čech) complex of the metric wedge sum is homotopy equivalent to the wedge sum of the Vietoris–Rips (resp. Čech) complexes. We also provide generalizations for certain more general metric gluings, namely, when two metric spaces are glued together along a common isometric subset.

One common metric space that appears in applications such as road networks [1], brain functional networks [8], and the cosmic web [29] is a *metric graph*, a structure where any two

points of the graph (not only vertices) are assigned a distance equal to the minimum length of a path from one point to the other. In this way, a metric graph encodes the proximity data of a network into the structure of a metric space. As a special case of our results, we show that the Vietoris–Rips complex of two metric graphs glued together along a sufficiently short common path is homotopy equivalent to the union of the Vietoris–Rips complexes. This enables us to determine the homotopy types of geometric thickenings of a large class of metric graphs, namely those that can be constructed iteratively via simple gluings.

The motivation for using Vietoris–Rips and Čech complexes in the context of data analysis is that these complexes can recover topological features of an unknown sample space underlying the data. Indeed, in [22, 25], it is shown that if the underlying space M is a closed Riemannian manifold, if scale parameter r is sufficiently small compared to the injectivity radius of M, and if a sample X is sufficiently close to M in the Gromov-Hausdorff distance, then the Vietoris–Rips complex of the sample X at scale r is homotopy equivalent to M. Analogously, the Nerve Theorem implies that the Čech complex (the nerve of all metric balls of radius r) of a similarly nice sample is homotopy equivalent to M for small r ([10] or [21, Corollary 4G.3]). In this paper, we identify the homotopy types of Vietoris–Rips and Čech complexes of certain metric graphs at all scale parameters r, not just at small scales.

Our paper builds on the authors' prior work characterizing the 1-dimensional intrinsic Čech persistence module associated to an arbitrary metric graph. Indeed, [20] shows that the 1-dimensional intrinsic Čech persistence diagram associated to a metric graph of genus g (i.e., the rank of the 1-dimensional homology of the graph) consists of the points $\left\{\left(0, \frac{\ell_i}{4}\right) : 1 \leq i \leq g\right\}$, where ℓ_i corresponds to the length of the i^{th} loop. In the case of the Vietoris–Rips complex, the results hold with the minor change that the persistence points are $\left\{\left(0, \frac{\ell_i}{6}\right) : 1 \leq i \leq g\right\}$. An extension of this work is [31], which studies the 1-dimensional persistence of geodesic spaces. In [3, 4], the authors show that the Vietoris–Rips or Čech complex of the circle obtains the homotopy types of the circle, the 3-sphere, the 5-sphere, ..., as the scale r increases, giving the persistent homology in all dimensions of a metric graph consisting of a single cycle. In this paper, we extend to a larger class of graphs: our results characterize the persistence profile, in any homological dimension, of Vietoris–Rips complexes of metric graphs that can be iteratively built by gluing trees and cycles together along short paths.

Our results on Vietoris–Rips and Čech complexes of metric gluings have implications for future algorithm development along the line of "topological decompositions". The computation time of homotopy, homology, and persistent homology depend on the size of the simplicial complex. It would be interesting to investigate if our Theorem 10 means that one can break a large metric graph into smaller pieces, perform computations on the simplicial complex of each piece, and then subsequently reassemble the results together. This has the potential to use less time and memory.

Outline. Section 2 introduces the necessary background and notation. Our main results on the Vietoris–Rips and Čech complexes of metric wedge sums and metric gluings are established in Section 3. In addition to proving homotopy equivalence in the wedge sum case, we show that the persistence module (for both Vietoris–Rips and Čech) of the wedge sum of the complexes is isomorphic to the persistence module of the complex for the wedge sum. We develop the necessary machinery to prove that the Vietoris–Rips complex of metric spaces glued together along a sufficiently short path is homotopy equivalent to the union of the Vietoris–Rips complexes. The machinery behind this proof technique does not hold in the analogous case for the Čech complex, and we provide an example illustrating why not. In Section 4, we describe families of metric graphs to which our results apply and furthermore

discuss those that we cannot yet characterize. In Section 5, we conclude with our overall goal of characterizing the persistent homology profiles of families of metric graphs.

2 Background

In this section, we recall the relevant background in the settings of simplicial complexes and metric spaces, including metric graphs. For a more comprehensive introduction of related concepts from algebraic topology, we refer the reader to [21], and to [23] and [19] for a combinatorial and computational treatment, respectively.

Simplicial complexes. An *abstract simplicial complex* K is a collection of finite subsets of some (possibly infinite) vertex set $V = V(K)$, such that if $\sigma \in K$ and $\tau \subseteq \sigma$, then $\tau \in K$. In this paper, we use the same symbol K to denote both the abstract simplicial complex and its geometric realization. For $V' \subseteq V$, we let $K[V']$ denote the induced simplicial complex on the vertex subset V'. If K and L are simplicial complexes with disjoint vertex sets $V(K)$ and $V(L)$, then their *join* $K * L$ is the simplicial complex whose vertex set is $V(K) \cup V(L)$, and whose set of simplices is $K * L = \{\sigma_K \cup \sigma_L \mid \sigma_K \in K \text{ and } \sigma_L \in L\}$ [23, Definition 2.16]. The *join* of two disjoint simplices $\sigma = \{x_0, \cdots, x_n\}$ and $\tau = \{y_0, \cdots, y_m\}$ is the simplex $\sigma \cup \tau := \{x_0, \cdots, x_n, y_0, \cdots, y_m\}$.

By an abuse of notation, a simplex $S \in K$ can be considered as either a single simplex, or as a simplicial complex $\{S' \mid S' \subseteq S\}$ with all subsets as faces. When taking joins, we use $\cup$ to denote that the result is a simplex, and we use $*$ to denote that the result is a simplicial complex. For example, for $a \in V(K)$ a vertex and $S \in K$ a simplex, we use the notation $a \cup S := \{a\} \cup S$ to denote the simplex formed by adding vertex a to S. We instead use $a * S := \{a \cup S' \mid S' \subseteq S\}$ to denote the associated simplicial complex. Similarly, for two simplices $\sigma, S \in K$, we use $\sigma \cup S$ to denote a simplex, and we instead use $\sigma * S := \{\sigma' \cup S' \mid \sigma' \subseteq \sigma,\ S' \subseteq S\}$ to denote the associated simplicial complex. We let $\dot{S}$ be the boundary simplicial complex $\dot{S} = \{S' \mid S' \subsetneq S\}$, and therefore $a * \dot{S} := \{a \cup S' \mid S' \subsetneq S\}$ and $\sigma * \dot{S} := \{\sigma' \cup S' \mid \sigma' \subseteq \sigma,\ S' \subsetneq S\}$ are simplicial complexes.

A simplicial complex K is equipped with the topology of a CW-complex [21]: a subset of the geometric realization of K is closed if and only if its intersection with each finite-dimensional skeleton is closed.

Simplicial collapse. Recall that if τ is a face of a simplex σ, then σ is said to be a *coface* of τ. Given a simplicial complex K and a maximal simplex $\sigma \in K$, we say that a face $\tau \subseteq \sigma$ is a *free face of* σ if σ is the unique maximal coface of τ in K. A *simplicial collapse* of K with respect to a pair (τ, σ), where τ is a free face of σ, is the removal of all simplices ρ such that $\tau \subseteq \rho \subseteq \sigma$. If $\dim(\sigma) = \dim(\tau) + 1$ then this is an *elementary simplicial collapse*. If L is obtained from a finite simplicial complex K via a sequence of simplicial collapses, then L is homotopy equivalent to K, denoted $L \simeq K$ [23, Proposition 6.14].

Metric spaces. Let (X, d) be a metric space, where X is a set equipped with a distance function d. Let $B(x, r) := \{y \in X \mid d(x, y) \leq r\}$ denote the closed ball with center $x \in X$ and radius $r \geq 0$. The *diameter* of X is $\operatorname{diam}(X) = \sup\{d(x, x') \mid x, x' \in X\}$. A *submetric space* of X is any set $X' \subseteq X$ with a distance function defined by restricting d to $X' \times X'$.

Vietoris–Rips and Čech complexes. We consider two types of simplicial complexes constructed from a metric space (X, d). These constructions depend on the choice of a scale

parameter $r \geq 0$. First, the *Vietoris–Rips complex* of X at scale $r \geq 0$ consists of all finite subsets of diameter at most r, that is, $\mathrm{VR}(X;r) = \{\text{finite } \sigma \subseteq X \mid \mathrm{diam}(\sigma) \leq r\}$. Second, for X a submetric space of X', we define the *ambient Čech complex* with vertex set X as $\check{\mathrm{C}}\mathrm{ech}(X, X'; r) := \{\text{finite } \sigma \subseteq X \mid \exists\, x' \in X' \text{ with } d(x, x') \leq r \;\forall\, x \in \sigma\}$. The set X is often called the set of "landmarks", and X' is called the set of "witnesses" [17]. This complex can equivalently be defined as the nerve of the balls $B_{X'}(x, r)$ in X' that are centered at points $x \in X$, that is, $\check{\mathrm{C}}\mathrm{ech}(X, X'; r) = \{\text{finite } \sigma \subseteq X \mid \bigcap_{x\in\sigma} B_{X'}(x, r) \neq \emptyset\}$. When $X = X'$, we denote the *(intrinsic) Čech complex* of X as $\check{\mathrm{C}}\mathrm{ech}(X; r) = \check{\mathrm{C}}\mathrm{ech}(X, X; r)$. Alternatively, the Čech complex can be defined with an open ball convention, and the Vietoris–Rips complex can be defined as $\mathrm{VR}(X; r) = \{\sigma \subseteq X \mid \mathrm{diam}(\sigma) < r\}$. Unless otherwise stated, all of our results hold for both the open and closed convention for Čech complexes, as well as for both the $<$ and $\leq$ convention on Vietoris–Rips complexes.

Persistent homology. For k a field, for $i \geq 0$ a homological dimension, and for Y a filtered topological space, we denote the *persistent homology* (or *persistence*) *module* of Y by $\mathrm{PH}_i(Y; k)$. Persistence modules form a category [16, Section 2.3], where morphisms are given by commutative diagrams.

Gluings of topological spaces. Let X and Y be two topological spaces that share a common subset $A = X \cap Y$. The *gluing* space $X \cup_A Y$ is formed by gluing X to Y along their common subspace A. More formally, let $\iota_X \colon A \to X$ and $\iota_Y \colon A \to Y$ denote the inclusion maps. Then the gluing space $X \cup_A Y$ is the quotient space of the disjoint union $X \coprod Y$ under the identification $\iota_X(a) \sim \iota_Y(a)$ for all $a \in A$. The gluing of two simplicial complexes along a common subcomplex is itself a simplicial complex.

Gluings of metric spaces. Following Definition 5.23 in [11], we define a way to glue two metric spaces along a closed subspace. Let X and Y be arbitrary metric spaces with closed subspaces $A_X \subset X$ and $A_Y \subset Y$. Let A be a metric space with isometries $\iota_X : A \to A_X$ and $\iota_Y : A \to A_Y$. Let $X \cup_A Y$ denote the quotient of the disjoint union of X and Y by the equivalence between A_X and A_Y, i.e., $X \cup_A Y = X \sqcup Y / \{\iota_X(a) \sim \iota_Y(a) \mid a \in A\}$. Then $X \cup_A Y$ is the *gluing of X and Y along A*. We define a metric on $X \cup_A Y$, which extends the metrics on X and Y:

$$d_{X\cup_A Y}(s,t) = \begin{cases} d_X(s,t) & \text{if } s,t \in X \\ d_Y(s,t) & \text{if } s,t \in Y \\ \inf_{a\in A} d_X(s,\iota_X(a)) + d_Y(\iota_Y(a),t) & \text{if } s \in X, t \in Y. \end{cases}$$

Lemma 5.24 of [11] shows that the gluing $(X \cup_A Y, d_{X\cup_A Y})$ of two metric spaces along common isometric closed subsets is itself a metric space. In this paper all of our metric gluings will be done in the case where $X \cap Y = A$ and the ι_X and ι_Y are identity maps. This definition of gluing metric spaces agrees with that of gluing their respective topological spaces, with the standard metric ball topology.

Pointed metric space and wedge sum. A *pointed metric space* is a metric space (X, d_X) with a distinguished basepoint $b_X \in X$. In the notation of metric gluings, given two pointed metric spaces (X, d_X) and (Y, d_Y), let $X \vee Y = X \cup_A Y$ where $A_X = \{b_X\}$ and $A_Y = \{b_Y\}$; we also refer to $X \vee Y$ as the *wedge sum* of X and Y. The gluing metric on $X \vee Y$ is the same as in the gluing of metric spaces above with $|A| = 1$.

Metric graphs. A graph G consists of set $V = \{v_i\}$ of vertices and a set $E = \{e_j\}$ of edges connecting the vertices. A graph G is a *metric graph* if each edge e_j is assigned a positive finite length l_j [11, 12, 24]. Under mild hypotheses[4], the graph G can be equipped with a natural metric d_G: the distance between any two points x and y (not necessarily vertices) in the metric graph is the infimum of the length of all paths between them.

Loops of a metric graph. A *loop* of a metric graph G is a continuous map $c : \mathbb{S}^1 \to G$. We also use the word *loop* to refer to the image of this map. Intuitively, elements of the singular 1-dimensional homology of G may be represented by collections of loops in G [21]. The *length* of a loop is the length of the image of the map c.

3 Homotopy equivalences for metric gluings

3.1 Homotopy lemmas for simplicial complexes

In this section, we present three lemmas that will be vital to our study of homotopy equivalences of simplicial complexes. We begin with a lemma proved by Barmak and Minian [7] regarding a sequence of elementary simplicial collapses between two simplicial complexes (Lemma 1). We then generalize this lemma in order to use it in the case where the simplicial collapses need not be elementary (Lemma 2). While these first two lemmas are relevant in the context of finite metric spaces, the third lemma will be useful when passing to arbitrary metric spaces. These three lemmas will later allow us to show that a complex built on a gluing is homotopy equivalent to the gluing of the complexes.

▶ **Lemma 1** (Lemma 3.9 from [7]). *Let L be a subcomplex of a finite simplicial complex K. Let T be a set of simplices of K which are not in L, and let a be a vertex of L which is contained in no simplex of T, but such that $a \cup S$ is a simplex of K for every $S \in T$. Finally, suppose that $K = L \cup \bigcup_{S \in T} \{S, a \cup S\}$. Then K is homotopy equivalent to L via a sequence of elementary simplicial collapses.*

In [7], Barmak and Mimian observe that there is an elementary simplicial collapse from K to L if there is a simplex S of K and a vertex a of K not in S such that $K = L \cup \{S, a \cup S\}$ and $L \cap (a * S) = a * \dot{S}$, where $\dot{S}$ denotes the boundary of S. Indeed, S is the free face of the elementary simplicial collapse, and the fact that $a \cup S$ is the unique maximal coface of S follows from $L \cap (a * S) = a * \dot{S}$ (which implies the intersection of L with S is the boundary of S). See Figure 1 (left) for an illustration.

It's not difficult to show that Barmak and Minian's observation can be made more general. In fact, there is a simplicial collapse from K to L if there is a simplex S of K and another simplex σ of K, disjoint from S, such that $K = L \cup \{\tau : S \subseteq \tau \subseteq \sigma \cup S\}$ and $L \cap (\sigma * S) = \sigma * \dot{S}$. Indeed, S is again the free face of the simplicial collapse, and the fact that $\sigma \cup S$ is the unique maximal coface of S in K follows from $L \cap (\sigma * S) = \sigma * \dot{S}$ (which implies the intersection of L with S is the boundary of S). See Figure 1 (right) for an illustration.

Our more general Lemma 2 will be used in the proof of Theorem 8 when we consider gluings along sets that are larger than just a single point.

[4] For every vertex, the lengths of the edges incident to that vertex are bounded away from zero [11, Section 1.9].

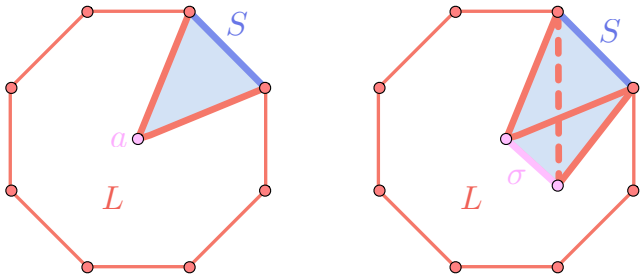


Figure 1 Examples of simplicial collapses when $T = \{S\}$ for Lemma 1 (left) and Lemma 2 (right).

▶ **Lemma 2** (Generalization of Lemma 1). *Let L be a subcomplex of a finite simplicial complex K, and let σ be a simplex in L. Suppose T is a set of simplices of K which are not in L and which are disjoint from σ, but such that $\sigma \cup S$ is a simplex of K for every $S \in T$. Finally, suppose $K = L \cup \bigcup_{\{S \in T\}} \{\tau \mid S \subseteq \tau \subseteq \sigma \cup S\}$. Then K is homotopy equivalent to L via a sequence of simplicial collapses.*

Proof. We mimic the proof of Lemma 3.9 in [7], except that we perform a sequence of simplicial collapses rather than elementary simplicial collapses. Order the elements S_1, S_2, $\ldots$, S_n of T in such a way that for every i, j with $i \leq j$, we have $|S_i| \leq |S_j|$. Define $K_i = L \cup \bigcup_{j=1}^{i} \{\tau \mid S_j \subseteq \tau \subseteq \sigma \cup S_j\}$ for $0 \leq i \leq n$. Let $S \subsetneq S_i$. If $S \in T$, then $\sigma \cup S \in K_{i-1}$ since $|S| < |S_i|$. If $S \notin T$, then $\sigma \cup S$ is in $L \subseteq K_{i-1}$. This proves that $K_{i-1} \cap (\sigma * S_i) = \sigma * \dot{S}_i$, and so S_i is the free face of a simplicial collapse from K_i to K_{i-1}. Then we are done since $K = K_n$ and $L = K_0$. ◀

The next lemma will be useful when passing from wedge sums or gluings of finite metric spaces to wedge sums or gluings of arbitrary metric spaces.

▶ **Lemma 3.** *Let K be a (possibly infinite) simplicial complex with vertex set V, and let L be a subcomplex also with vertex set V. Suppose that for every finite $V_0 \subseteq V$, there exists a finite subset V_1 with $V_0 \subseteq V_1 \subseteq V$ such that the inclusion $L[V_1] \hookrightarrow K[V_1]$ is a homotopy equivalence. It then follows that the inclusion map $\iota \colon L \hookrightarrow K$ is a homotopy equivalence.*

Proof. We will use a compactness argument to show that the induced mapping on homotopy groups $\iota_* \colon \pi_k(L, b) \to \pi_k(K, b)$ is an isomorphism for all k and for any basepoint b in the geometric realization of L. The conclusion then follows from Whitehead's theorem [21, Theorem 4.5].

First, suppose we have a based map $f \colon \mathbb{S}^k \to K$ where $\mathbb{S}^k$ is the k-dimensional sphere. Since f is continuous and $\mathbb{S}^k$ is compact, it follows that $f(\mathbb{S}^k)$ is compact in K. Then by [21, Proposition A.1] we know that $f(\mathbb{S}^k)$ is contained in a finite subcomplex of K. Therefore, there exists a finite subset $V_0 \subseteq V$ so that f factors through $K[V_0] \subseteq K$. By assumption, there exists a finite subset V_1 with $V_0 \subseteq V_1 \subseteq V$ such that the inclusion $\iota_1 \colon L[V_1] \hookrightarrow K[V_1]$ is a homotopy equivalence. Thus, we can find a based map $\widetilde{f} \colon \mathbb{S}^k \to L[V_1]$ such that $[\iota_1 \widetilde{f}] = [f] \in \pi_k(K[V_1], b)$ and hence $[\iota \widetilde{f}] = [f] \in \pi_k(K, b)$. This proves that ι_* is surjective.

Next, suppose that $f \colon \mathbb{S}^k \to L$ is a based map such that $\iota f \colon \mathbb{S}^k \to K$ is null-homotopic. Let $F \colon \mathbb{B}^{k+1} \to K$ be a null-homotopy between ιf and the constant map, where $\mathbb{B}^{k+1}$ is the $(k+1)$-dimensional ball. By compactness of S^k and B^{k+1}, we can find a finite subset $V_0 \subseteq V$ such that f factors through $L[V_0]$ and F factors through $K[V_0]$. By assumption, there exists a finite subset V_1 with $V_0 \subseteq V_1 \subseteq V$ such that the inclusion $\iota_1 \colon L[V_1] \hookrightarrow K[V_1]$ is a homotopy equivalence. Note that $\iota_1 f \colon \mathbb{S}^k \to K[V_1]$ is null-homotopic via F, and since the inclusion ι_1 is a homotopy equivalence, it follows that f is null-homotopic, and thus ι_* is injective. ◀

3.2 Vietoris–Rips and Čech complexes of wedge sums

As a warm-up, we first show in this subsection that the Vietoris–Rips complex of a metric wedge sum (i.e, gluing along a single point) is homotopy equivalent to the wedge sum of the Vietoris–Rips complexes, and similarly for Čech complexes. In the next subsection, Proposition 4 will be extended in Corollary 9 and Theorem 10 to gluings of metric spaces and to gluings of metric graphs along short paths, respectively. Intuitively, such results allow us to characterize the topology of a bigger space via the topology of smaller individual pieces.

Given pointed metric spaces X and Y, we use the symbol $b \in X \vee Y$ to denote the point corresponding to the identified distinguished basepoints $b_X \in X$ and $b_Y \in Y$.

▶ **Proposition 4.** *For X and Y pointed metric spaces and $r > 0$, we have the homotopy equivalence* $\mathrm{VR}(X;r) \vee \mathrm{VR}(Y;r) \xhookrightarrow{\sim} \mathrm{VR}(X \vee Y;r)$.

Proof. We first consider the case where X and Y are finite. We apply Lemma 1 with $L = \mathrm{VR}(X;r) \vee \mathrm{VR}(Y;r)$, with $K = \mathrm{VR}(X \vee Y;r)$, with $T = \{\sigma \in K \setminus L \mid b \notin \sigma\}$, and with basepoint $b \in X \vee Y$ serving the role as a. It is easy to check the conditions on K, L and T required by Lemma 1 are satisfied. Furthermore, if $\sigma \in T$, then at least one vertex of $X \setminus \{b_X\}$ and one vertex of $Y \setminus \{b_Y\}$ are in σ. Hence $\mathrm{diam}(\sigma \cup b) \leq r$ and $\sigma \cup b$ is a simplex of K. Since $K = L \cup \bigcup_{\sigma \in T} \{\sigma, \sigma \cup b\}$, Lemma 1 implies $L \simeq K$.

Now let X and Y be arbitrary (possibly infinite) pointed metric spaces. For finite subsets $X_0 \subseteq X$ and $Y_0 \subseteq Y$ with $b_X \in X_0$ and $b_Y \in Y_0$, the finite case guarantees that $\mathrm{VR}(X_0;r) \vee \mathrm{VR}(Y_0;r) \simeq \mathrm{VR}(X_0 \vee Y_0;r)$. Therefore, we can apply Lemma 3 with $L = \mathrm{VR}(X;r) \vee \mathrm{VR}(Y;r)$ and $K = \mathrm{VR}(X \vee Y;r)$. ◀

Proposition 4, in the case of finite metric spaces, is also obtained in [26].

▶ **Corollary 5.** *Let X and Y be pointed metric spaces. For any homological dimension $i \geq 0$ and field k, the persistence modules* $\mathrm{PH}_i(\mathrm{VR}(X;r) \vee \mathrm{VR}(Y;r);k)$ *and* $\mathrm{PH}_i(\mathrm{VR}(X \vee Y;r);k)$ *are isomorphic.*

See the full version of this paper [5] for the proof of Corollary 5.

For a submetric space $X \subseteq X'$, let $\check{\mathrm{C}}\mathrm{ech}(X, X';r)$ be the ambient Čech complex with landmark set X and witness set X'. Note that if $X \subseteq X'$ and $Y \subseteq Y'$ are pointed with $b_X = b_{X'}$ and $b_Y = b_{Y'}$, then $X \vee Y$ is a submetric space of $X' \vee Y'$.

▶ **Proposition 6.** *For $X \subseteq X'$ and $Y \subseteq Y'$ pointed metric spaces and $r > 0$, we have the homotopy equivalence* $\check{\mathrm{C}}\mathrm{ech}(X, X';r) \vee \check{\mathrm{C}}\mathrm{ech}(Y, Y';r) \xhookrightarrow{\sim} \check{\mathrm{C}}\mathrm{ech}(X \vee Y, X' \vee Y';r)$.

The proof of Proposition 6 is in the full version of this paper. It proceeds similarly to the proof of Proposition 4, except applying Lemma 1 with $L = \check{\mathrm{C}}\mathrm{ech}(X, X';r) \vee \check{\mathrm{C}}\mathrm{ech}(Y, Y';r)$, $K = \check{\mathrm{C}}\mathrm{ech}(X \vee Y, X' \vee Y';r)$, and $T = \{\sigma \in K \setminus L \mid b \notin \sigma\}$.

▶ **Corollary 7.** *Let $X \subseteq X'$ and $Y \subseteq Y'$ be pointed metric spaces. For any homological dimension $i \geq 0$ and field k, the persistence modules* $\mathrm{PH}_i(\check{\mathrm{C}}\mathrm{ech}(X, X';r) \vee \check{\mathrm{C}}\mathrm{ech}(Y, Y';r))$ *and* $\mathrm{PH}_i(\check{\mathrm{C}}\mathrm{ech}(X \vee Y, X' \vee Y';r))$ *are isomorphic.*

3.3 Vietoris–Rips complexes of set-wise gluings

We now develop the machinery necessary to prove, in Theorem 10, that the Vietoris–Rips complex of two metric graphs glued together along a sufficiently short path is homotopy equivalent to the union of the Vietoris–Rips complexes. First, we prove a more general result for arbitrary metric spaces that intersect in a sufficiently small space.

▶ **Theorem 8.** *Let X and Y be metric spaces with $X \cap Y = A$, where A is a closed subspace of X and Y, and let $r > 0$. Consider $X \cup_A Y$, the metric gluing of X and Y along the intersection A. Suppose that if $\operatorname{diam}(S_X \cup S_Y) \leq r$ for some $\emptyset \neq S_X \subseteq X \setminus A$ and $\emptyset \neq S_Y \subseteq Y \setminus A$, then there is a unique maximal nonempty subset $\sigma \subseteq A$ such that $\operatorname{diam}(S_X \cup S_Y \cup \sigma) \leq r$. Then $\operatorname{VR}(X \cup_A Y; r) \simeq \operatorname{VR}(X; r) \cup_{\operatorname{VR}(A;r)} \operatorname{VR}(Y; r)$. Hence if $\operatorname{VR}(A; r)$ is contractible, then $\operatorname{VR}(X \cup_A Y; r) \simeq \operatorname{VR}(X; r) \vee \operatorname{VR}(Y; r)$.*

Proof. We first restrict our attention to the case when X and Y (and hence A) are finite. Let $n = |A|$. Order the nonempty subsets $\sigma_1, \sigma_2, \ldots, \sigma_{2^n-1}$ of A so that for every i, j with $i \leq j$, we have $|\sigma_i| \geq |\sigma_j|$. For $i = 1, 2, \ldots, 2^n - 1$, let T_i be the set of all simplices of the form $S_X \cup S_Y$ such that

- $\emptyset \neq S_X \subseteq X \setminus A$ and $\emptyset \neq S_Y \subseteq Y \setminus A$,
- $\operatorname{diam}(S_X \cup S_Y) \leq r$, and
- σ_i is the maximal nonempty subset of A satisfying $\operatorname{diam}(S_X \cup S_Y \cup \sigma_i) \leq r$.

Let $L_0 = \operatorname{VR}(X; r) \cup_{\operatorname{VR}(A;r)} \operatorname{VR}(Y; r)$. We apply Lemma 2 repeatedly to obtain

$$\begin{aligned} L_0 \simeq & L_0 \cup_{\{S \in T_1\}} \{\tau \mid S \subseteq \tau \subseteq \sigma_1 \cup S\} =: L_1 \\ \simeq & L_1 \cup_{\{S \in T_2\}} \{\tau \mid S \subseteq \tau \subseteq \sigma_2 \cup S\} =: L_2 \\ & \vdots \\ \simeq & L_{2^n-3} \cup_{\{S \in T_{2^n-2}\}} \{\tau \mid S \subseteq \tau \subseteq \sigma_{2^n-2} \cup S\} =: L_{2^n-2} \\ \simeq & L_{2^n-2} \cup_{\{S \in T_{2^n-1}\}} \{\tau \mid S \subseteq \tau \subseteq \sigma_{2^n-1} \cup S\} =: L_{2^n-1}. \end{aligned}$$

The fact that each L_j is a simplicial complex follows since if $S_X \cup S_Y \in T_j$ and $\emptyset \neq S'_X \subseteq S_X$ and $\emptyset \neq S'_Y \subseteq S_Y$, then we have $S'_X \cup S'_Y \in T_i$ for some $i \leq j$ (meaning $\sigma_j \subseteq \sigma_i$). For each $j = 1, \ldots, 2^n - 1$, we set $K = L_j$, $L = L_{j-1}$, $T = T_j$, and $\sigma = \sigma_j$ and apply Lemma 2 to get that $L_j \simeq L_{j-1}$ (This works even when $T_j = \emptyset$, in which case $L_j = L_{j-1}$).

To complete the proof of the finite case it suffices to show $L_{2^n-1} = \operatorname{VR}(X \cup_A Y; r)$. This is because any simplex $\tau \in \operatorname{VR}(X \cup_A Y; r) \setminus L_0$ is necessarily of the form $\tau = S_X \cup S_Y \cup \sigma$, with $\emptyset \neq S_X \subseteq X \setminus A$, with $\emptyset \neq S_Y \subseteq Y \setminus A$, with $\sigma \subseteq A$, and with $\operatorname{diam}(S_X \cup S_Y \cup \sigma) \leq r$. By assumption, there exists a unique maximal non-empty set $\sigma_j \subseteq A$ such that $\operatorname{diam}(S_X \cup S_Y \cup \sigma_j) \leq r$. Since σ_j is unique, we have that $\sigma \subseteq \sigma_j$. Therefore $S_X \cup S_Y \in T_j$, and τ will be added to L_j since $S_X \cup S_Y \subseteq \tau \subseteq S_X \cup S_Y \cup \sigma_j$. Hence $\tau \in L_{2^n-1}$ and so $L_{2^n-1} = \operatorname{VR}(X \cup_A Y; r)$.

Now let X and Y be arbitrary metric spaces. Note that for any finite subsets $X_0 \subseteq X$ and $Y_0 \subseteq Y$ with $A_0 = X_0 \cap Y_0 \neq \emptyset$, we have $\operatorname{VR}(X_0; r) \cup_{\operatorname{VR}(A_0;r)} \operatorname{VR}(Y_0; r) \simeq \operatorname{VR}(X_0 \cup_{A_0} Y_0; r)$ by the finite case. Hence we can apply Lemma 3 with $L = \operatorname{VR}(X; r) \cup_{\operatorname{VR}(A;r)} \operatorname{VR}(Y; r)$ and $K = \operatorname{VR}(X \cup_A Y; r)$ to complete the proof. ◀

▶ **Corollary 9.** *Let X and Y be metric spaces with $X \cap Y = A$, where A is a closed subspace of X and Y, and $X \cup_A Y$ is their metric gluing along A. Let $r > 0$, and suppose $\operatorname{diam}(A) \leq r$. Then $\operatorname{VR}(X \cup_A Y; r) \simeq \operatorname{VR}(X; r) \vee \operatorname{VR}(Y; r)$.*

Proof. The fact that $\operatorname{diam}(A) \leq r$ implies that if $\operatorname{diam}(S_X \cup S_Y) \leq r$ for some $S_X \subseteq X \setminus A$ and $S_Y \subseteq Y \setminus A$, there is a unique maximal nonempty subset $\sigma \subseteq A$ such that $\operatorname{diam}(S_X \cup S_Y \cup \sigma) \leq r$. Indeed, the set of all such $\sigma \subseteq A$ satisfying $\operatorname{diam}(S_X \cup S_Y \cup \sigma) \leq r$ is closed under unions since $\operatorname{diam}(A) \leq r$, and hence there will be a unique maximal σ. The definition of the metric on $X \cup_A Y$ implies that $\sigma \neq \emptyset$. The claim now follows from Theorem 8 and from the fact that $\operatorname{VR}(A; r)$ is contractible. ◀

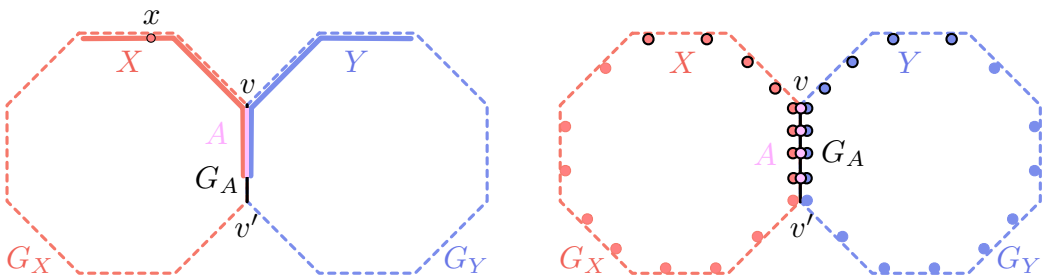


Figure 2 Illustration of Theorem 10 on metric graph gluings and both infinite (left) and finite (right) subsets thereof. The metric graphs G_X and G_Y are shown with thin, dotted red and blue lines respectively; X and Y are shown in the infinite case with thick, solid red and blue lines respectively; G_A corresponds to the black solid line while A corresponds to the pink solid line. The finite subset case uses the same color scheme.

Note that if A is a single point (i.e. $|A| = 1$), then $X \cup_A Y$ is the same as $X \vee Y$. Therefore, Proposition 4 is a special case of Corollary 9.

The setup of the following theorem regarding metric graph gluings is illustrated in Figure 2. In a graph, the *degree* of a vertex without self-loops is the number of incident edges to that vertex. A *path graph* (or simply a path) is one in which the n vertices can be ordered, and in which the $n-1$ edges connect pairs of successive vertices.

▶ **Theorem 10.** *Let $G = G_X \cup_{G_A} G_Y$ be a metric graph, where $G_A = G_X \cap G_Y$ is a closed metric subgraph of the metric graphs G_X and G_Y. Suppose furthermore that G_A is a path, and that each vertex of G_A besides the two endpoints has degree 2 not only as a vertex in G_A, but also as a vertex in G. Suppose the length of G_A is at most $\frac{\ell}{3}$, where any simple loop in G that goes through G_A has length at least ℓ. Let $X \subseteq G_X$ and $Y \subseteq G_Y$ be arbitrary subsets such that $X \cap G_Y = Y \cap G_X = X \cap Y := A$. Then $\mathrm{VR}(X \cup_A Y; r) \simeq \mathrm{VR}(X; r) \cup_{\mathrm{VR}(A;r)} \mathrm{VR}(Y; r)$ for all $r > 0$. Hence if $\mathrm{VR}(A; r)$ is contractible, then $\mathrm{VR}(X \cup_A Y; r) \simeq \mathrm{VR}(X; r) \vee \mathrm{VR}(Y; r)$.*

Proof. Let the length of G_A be $\alpha \leq \frac{\ell}{3}$. If $r \geq \alpha$, then the conclusion follows from Corollary 9.

For the case $r < \alpha \leq \frac{\ell}{3}$, let v and v' be the endpoints of the path G_A. We claim that no point $z \in (X \cup_A Y) \setminus A$ is within distance r of both v and v'. Indeed, if there were such a point $z \in (X \cup_A Y) \setminus A$ satisfying $d(z, v) \leq r$ and $d(z, v') \leq r$, then we could produce a homologically non-trivial loop through the gluing path G_A that is shorter than ℓ, giving a contradiction. It follows that if $\mathrm{diam}(S_X \cup S_Y) \leq r$ for some $S_X \subseteq X \setminus A$ and $S_Y \subseteq Y \setminus A$, then there is a unique maximal nonempty set $\sigma \subseteq A$ such that $\mathrm{diam}(S_X \cup S_Y \cup \sigma) \leq r$. To be more explicit, one can show that this maximal set is $\sigma = A \cap \bigcap_{z \in S_X \cup S_Y} B(z, r)$; it is nonempty since it contains either v or v'. Hence, Theorem 8 implies that $\mathrm{VR}(X \cup_A Y; r) \simeq \mathrm{VR}(X; r) \cup_{\mathrm{VR}(A;r)} \mathrm{VR}(Y; r)$. ◀

▶ **Corollary 11.** *Let G, G_X, G_Y, G_A, X, Y, and A satisfy the same hypotheses as in the statement of Theorem 10. Suppose furthermore that $\mathrm{VR}(A; r)$ is contractible for all $r > 0$. Then for any homological dimension $i \geq 0$ and field k, the persistence modules $\mathrm{PH}_i(\mathrm{VR}(X; r) \vee \mathrm{VR}(Y; r); k)$ and $\mathrm{PH}_i(\mathrm{VR}(X \cup_A Y; r))$ are isomorphic.*

Remarks on set-wise gluings in the Čech case. It seems natural to ask if our results in Section 3.3 extend to Čech complexes. In other words, is it necessarily the case that $\check{\mathrm{C}}\mathrm{ech}(X; r) \cup_{\check{\mathrm{C}}\mathrm{ech}(A;r)} \check{\mathrm{C}}\mathrm{ech}(Y; r) \simeq \check{\mathrm{C}}\mathrm{ech}(X \cup_A Y; r)$, where X, Y and A are as described in Theorem 10? Interestingly, while the desired result may hold true, the arguments of Section 3.3 do not all directly transfer to the Čech case. In particular, Theorem 8 can

be extended to the Čech case by replacing the condition $\operatorname{diam}(S_X \cup S_Y \cup \sigma) \leq r$ with $\bigcap_{z \in S_X \cup S_Y \cup \sigma} B(z;r) \neq \emptyset$. However, the arguments for Corollary 9 do not transfer to the Čech case no matter how small the size of the gluing portion A is, which subsequently makes it hard to adapt the strategy behind Theorem 10 to the Čech case. An example to illustrate this is given in the full version. This suggests that a different technique needs to be developed in order to show an analog of Theorem 10 for the Čech setting (if such an analog holds).

4 Applicability to certain families of graphs

The results in Section 3 provide a mechanism to compute the homotopy types and persistent homology of Vietoris–Rips complexes of metric spaces built from gluing together simpler ones. For the sake of brevity, if the results of Section 3 can be used to completely describe the homotopy types and persistence module of the Vietoris–Rips complex of metric space X, then we will simply say that space X can be *characterized*. Figure 3 shows examples of two metric graphs that can be characterized (a and b) and two that cannot (c and d). In Section 4.1, we describe some families of metric spaces that can be characterized, and in Section 4.2, we discuss graphs (c) and (d).

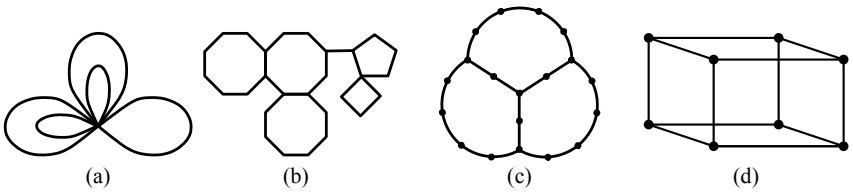


Figure 3 Graphs (a) and (b) can be characterized while (c) and (d) cannot.

4.1 Families of graphs

In this section we consider finite metric spaces and metric graphs that can be understood using the results in this paper. Examples of finite metric spaces whose Vietoris–Rips complexes are well-understood include the vertex sets of dismantlable graphs (defined below) and vertex sets of single cycles [2]. Examples of metric graphs whose Vietoris–Rips complexes are well-understood include trees and single cycles [3].

Let G be a graph with vertex set V and with all edges of length one.[5] The vertex set V is a metric space equipped with the shortest path metric. We say that a vertex $v \in V$ is *dominated* by $u \in V$ if v is connected to u, and if each neighbor of v is also a neighbor of u. We say that a graph is *dismantlable* if we can iteratively remove dominated vertices from G in order to obtain the graph with a single vertex. Note that if v is dominated by u, then v is dominated by u in the 1-skeleton of $\mathrm{VR}(V;r)$ for all $r \geq 1$. It follows from the theory of folds, elementary simplicial collapses, or LC reductions [6, 9, 27] that if G is dismantlable, then $\mathrm{VR}(V;r)$ is contractible for all $r \geq 1$. Examples of dismantlable graphs include trees, chordal graphs, and unit disk graphs of sufficiently dense samplings of convex sets in the plane [25, Lemma 2.1]. We will also need the notion of a *k-cycle* graph, a simple cycle with k vertices and k edges. The following proposition specifies a family of finite metric spaces that can be characterized using the results in this paper.

[5] We make this assumption for simplicity's sake, even though it can be relaxed.

▶ **Proposition 12.** *Let G be a finite graph, with each edge of length one, that can be obtained from a vertex by iteratively attaching (i) a dismantlable graph or (ii) a k-cycle graph along a vertex or along a single edge. Let V be the vertex set of the graph G. Then we have $\mathrm{VR}(V;r) \simeq \bigvee_{i=1}^{n} \mathrm{VR}(V(C_{k_i});r)$ for $r \geq 1$, where n is the number of times operation (ii) is performed, k_i are the corresponding cycle lengths, and $V(C_{k_i})$ is the vertex set of a k_i-cycle.*

Proof. It suffices to show that an operation of type (i) does not change the homotopy type of the Vietoris–Rips complex of the vertex set, and that an operation of type (ii) has the effect up to homotopy of taking a wedge sum with $\mathrm{VR}(V(C_k);r)$. The former follows from applying Corollary 9 (or alternatively Theorem 10), as the Vietoris–Rips complex of the vertex set of a dismantlable graph is contractible for all $r \geq 1$, and the latter also follows from Corollary 9 (or alternatively Theorem 10). ◀

The iterative procedure outlined in Proposition 12 can be used to obtain some recognizable families of graphs. Examples include trees and wedge sums of cycles (Figure 3(a)). More complicated are *polygon trees* [18] in which cycles are iteratively attached along a single edge. Graph (b) in Figure 3 is an example that is built by using both (i) and (ii).

A similar procedure is possible for metric graphs, except that we must replace arbitrary dismantlable graphs with the specific case of trees.[6] Note that any tree, T, is obtained by starting with a single vertex and iteratively taking the wedge sum with a single edge. This implies that $\mathrm{VR}(T;r)$ is contractible; thus, the persistent homology filtration of any tree is trivial, a result that is also established in [14, 28].

▶ **Proposition 13.** *Let G be a metric graph, with each edge of length one, that can be obtained from a vertex by iteratively attaching (i) an edge along a vertex or (ii) a k-cycle graph along a vertex or a single edge. Then we have $\mathrm{VR}(G;r) \simeq \bigvee_{i=1}^{n} \mathrm{VR}(C_{k_i};r)$ for $r \geq 1$, where n is the number of times operation (ii) is performed, k_i are the corresponding cycle lengths, and C_{k_i} is a loop of length k_i.*

Proof. The proof is analogous to that of Proposition 12. ◀

Proposition 13 can be generalized to allow for arbitrary edge lengths, as long as the conditions of Theorem 10 hold (see the full version for further discussion and an example).

4.2 Obstructions to using our results

When gluing two metric graphs, G_1 and G_2, the most restrictive requirement is that the gluing path must be a simple path with all vertices except the endpoints having degree 2. This requirement is what disallows the configuration (c) in Figure 3. Notice that every pair of shortest cycles shares only a simple path of length 2. However, once one pair is glued, the third must be glued along a path of length 4 which traverses both of the two other cycles. This path includes a vertex of degree 3 in its interior, meaning that Theorem 10 is not applicable. Moreover, when $r < 4$, (where 4 is the diameter of the gluing set), then Corollary 9 is also not applicable. Nevertheless, we can computationally verify that for (c), the homology of the Vietoris–Rips complex is still the direct sum of the homology groups of the component cycles (where $\mathrm{VR}(V(C_9);r) \simeq S^1$ for $r = 1$ or 2, and where $\mathrm{VR}(V(C_9);3) \simeq S^2 \vee S^2$ [4]). In

[6] The case of C_3, a cyclic graph with three unit-length edges, is instructive. Since C_3 is dismantlable we have that $\mathrm{VR}(V(C_3);r)$ is contractible for any $r \geq 1$. But since the metric graph C_3 is isometric to a circle of circumference 3, it follows from [3] that $\mathrm{VR}(C_3;r)$ is not contractible for $0 < r < \frac{3}{2}$.

light of this example, in future work, we hope to extend the results in this paper to gluing metric graphs along admissible isometric trees (a generalization of isometric simple paths).

The final graph (d) in Figure 3, the cube graph, is another case for which Theorem 10 is not applicable. However, unlike example (c) above, we cannot compute the homology of the vertex set of the cube as the direct sum of the homology groups of smaller component pieces. Indeed, if V is the vertex set of the cube with each edge of length one, then $\dim(H_3(\mathrm{VR}(V;2))) = 1$ since $\mathrm{VR}(V;2)$ is homotopy equivalent to the 3-sphere. However, this graph is the union of five cycles of length four, and the Vietoris–Rips complex of the vertex set of a cycle of length four never has any 3-dimensional homology.

5 Discussion

We have shown that the wedge sum of Vietoris–Rips (resp. Čech) complexes is homotopy equivalent to the corresponding complex for the metric wedge sum, and generalized this result in the case of Vietoris–Rips complexes for certain metric space gluings. Our ultimate goal is to understand to the greatest extent possible the topological structure of large classes of metric graphs via persistent homology. Building on previous work in [3] and [20], the results in this paper constitute another important step toward this goal by providing a characterization of the persistence profiles of metric graphs obtainable via certain types of metric gluing. Many interesting questions remain for future research.

Gluing beyond a single path. We are interested in studying metric graphs obtainable via metric gluings other than along single paths of degree 2, such as gluing along a tree or self-gluing. For the case of gluings along a tree, the gluing graph may have vertices of degree greater than 2. Examples include gluing four square graphs together into a larger square with a degree 4 node in the center. Moreover, the techniques of our paper do not allow one to analyze self-gluings such as forming an n-cycle C_n from a path of length n.

Generative models for metric graphs. Our results can be considered as providing a generative model for metric graphs, where we specify a particular metric gluing rule for which we have a clear understanding of its effects on persistent homology. Expanding the list of metric gluing rules would in turn lead to a larger collection of generative models.

Approximations of persistent homology profiles. A particular metric graph that arises from data in practice may not directly correspond to an existing generative model. However, we may still be able to approximate its persistent homology profile via stability results (e.g. [15, 30]) by demonstrating close proximity between its metric and a known one.

References

1 Mridul Aanjaneya, Frederic Chazal, Daniel Chen, Marc Glisse, Leonidas Guibas, and Dmitry Morozon. Metric graph reconstruction from noisy data. *International Journal of Computational Geometry and Applications*, 22(04):305–325, 2012.

2 Michał Adamaszek. Clique complexes and graph powers. *Israel Journal of Mathematics*, 196(1):295–319, 2013.

3 Michał Adamaszek and Henry Adams. The Vietoris–Rips complexes of a circle. *Pacific Journal of Mathematics*, 290:1–40, 2017.

4 Michał Adamaszek, Henry Adams, Florian Frick, Chris Peterson, and Corrine Previte-Johnson. Nerve complexes of circular arcs. *Discrete & Computational Geometry*, 56(2):251–273, 2016.

5 Michał Adamaszek, Henry Adams, Ellen Gasparovic, Marix Gommel, Emilie Purvine, Radmila Sazdanovic, Bei Wang, Yusu Wang, and Lori Ziegelmeier. Vietoris-Rips and Čech complexes of metric gluings. *arXiv:1712.06224*, 2018.

6 Eric Babson and Dmitry N. Kozlov. Complexes of graph homomorphisms. *Israel Journal of Mathematics*, 152(1):285–312, 2006.

7 Jonathan Ariel Barmak and Elias Gabriel Minian. Simple homotopy types and finite spaces. *Advances in Mathematics*, 218:87–104, 2008.

8 Bharat Biswal, F. Zerrin Yetkin, Victor M. Haughton, and James S. Hyde. Functional connectivity in the motor cortex of resting human brain using echo-planar MRI. *Magnetic Resonance in Medicine*, 34:537–541, 1995.

9 Anders Björner. Topological methods. *Handbook of Combinatorics*, 2:1819–1872, 1995.

10 Karol Borsuk. On the imbedding of systems of compacta in simplicial complexes. *Fundamenta Mathematicae*, 35(1):217–234, 1948.

11 Martin R. Bridson and André Haefliger. *Metric Spaces of Non-Positive Curvature*. Springer-Verlag, 1999.

12 Dmitri Burago, Yuri Burago, and Sergei Ivanov. *A course in metric geometry*, volume 33. American Mathematical Society Providence, RI, 2001.

13 Gunnar Carlsson. Topology and data. *Bulletin of the American Mathematical Society*, 46(2):255–308, 2009.

14 Joseph Minhow Chan, Gunnar Carlsson, and Raul Rabadan. Topology of viral evolution. *Proceedings of the National Academy of Sciences*, 110(46):18566–18571, 2013.

15 Frédéric Chazal, David Cohen-Steiner, Leonidas J. Guibas, Facundo Mémoli, and Steve Y. Oudot. Gromov-Hausdorff stable signatures for shapes using persistence. *Computer Graphics Forum*, 28(5):1393–1403, 2009.

16 Frédéric Chazal, Vin De Silva, Marc Glisse, and Steve Oudot. *The structure and stability of persistence modules*. Springer, 2016.

17 Frédéric Chazal, Vin de Silva, and Steve Oudot. Persistence stability for geometric complexes. *Geometriae Dedicata*, pages 1–22, 2013.

18 Fengming Dong, Khee-Meng Koh, and Kee L Teo. *Chromatic Polynomials and Chromaticity of Graphs*. World Scientific, 2005.

19 Herbert Edelsbrunner and John L Harer. *Computational Topology: An Introduction*. American Mathematical Society, 2010.

20 Ellen Gasparovic, Maria Gommel, Emilie Purvine, Radmila Sazdanovic, Bei Wang, Yusu Wang, and Lori Ziegelmeier. A complete characterization of the one-dimensional intrinsic Čech persistence diagrams for metric graphs. In *Research in Computational Topology*, 2018.

21 Allen Hatcher. *Algebraic Topology*. Cambridge University Press, 2002.

22 Jean-Claude Hausmann. On the Vietoris–Rips complexes and a cohomology theory for metric spaces. *Annals of Mathematics Studies*, 138:175–188, 1995.

23 Dmitry N. Kozlov. *Combinatorial Algebraic Topology*, volume 21 of *Algorithms and Computation in Mathematics*. Springer-Verlag, 2008.

24 Peter Kuchment. Quantum graphs I. Some basic structures. *Waves in Random Media*, 14(1):S107–S128, 2004.

25 Janko Latschev. Vietoris–Rips complexes of metric spaces near a closed Riemannian manifold. *Archiv der Mathematik*, 77(6):522–528, 2001.

26 Michael Lesnick, Raul Rabadan, and Daniel Rosenbloom. Quantifying genetic innovation: Mathematical foundations for the topological study of reticulate evolution. In preparation, 2018.

27 Jiří Matoušek. LC reductions yield isomorphic simplicial complexes. *Contributions to Discrete Mathematics*, 3(2), 2008.

28 Steve Oudot and Elchana Solomon. Barcode embeddings, persistence distortion, and inverse problems for metric graphs. *arXiv: 1712.03630*, 2017.

29 Thierry Sousbie, Christophe Pichon, and Hajime Kawahara. The persistent cosmic web and its filamentary structure – II. Illustrations. *Monthly Notices of the Royal Astronomical Society*, 414(1):384–403, 2011.

30 Katharine Turner. Generalizations of the Rips filtration for quasi-metric spaces with persistent homology stability results. *arXiv:1608.00365*, 2016.

31 Žiga Virk. 1-dimensional intrinsic persistence of geodesic spaces. *arXiv:1709.05164*, 2017.

Improved Dynamic Geodesic Nearest Neighbor Searching in a Simple Polygon

Pankaj K. Agarwal[1]
Department of Computer Science, Duke University,
Durham, NC 27708, USA
pankaj@cs.duke.edu

Lars Arge[2]
MADALGO, Aarhus University,
Aarhus, Denmark
large@cs.au.dk

Frank Staals[3]
Dept. of Information and Computing Sciences, Utrecht University,
Utrecht, The Netherlands
f.staals@uu.nl

Abstract

We present an efficient dynamic data structure that supports geodesic nearest neighbor queries for a set S of point sites in a static simple polygon P. Our data structure allows us to insert a new site in S, delete a site from S, and ask for the site in S closest to an arbitrary query point $q \in P$. All distances are measured using the geodesic distance, that is, the length of the shortest path that is completely contained in P. Our data structure achieves polylogarithmic update and query times, and uses $O(n \log^3 n \log m + m)$ space, where n is the number of sites in S and m is the number of vertices in P. The crucial ingredient in our data structure is an implicit representation of a vertical shallow cutting of the geodesic distance functions. We show that such an implicit representation exists, and that we can compute it efficiently.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases data structure, simple polygon, geodesic distance, nearest neighbor searching, shallow cutting

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.4

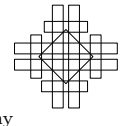
Related Version A full version of this paper is available at `http://arxiv.org/abs/1803.05765`

1 Introduction

Nearest neighbor searching is a classic problem in computational geometry in which we are given a set of point *sites* S, and we wish to preprocess these points such that for a query point q, we can efficiently find the site $s \in S$ closest to q. We consider the case where S is a *dynamic* set of points inside a simple polygon P. That is, we may insert a new site into S or delete an existing one. We measure the distance between two points p and q by the

[1] P.A. is supported by NSF under grants CCF-15-13816, CCF-15-46392, and IIS-14-08846, by ARO under grant W911NF-15-1-0408, and by the U.S.–Israel Binational Science Foundation under grant 2012/229.
[2] L.A. was supported by the Danish National Research Foundation under grant nr. DNRF84
[3] F.S. was supported by the Netherlands Organisation for Scientific Research under project no. 612.001.651.

34th International Symposium on Computational Geometry (SoCG 2018).
Editors: Bettina Speckmann and Csaba D. Tóth; Article No. 4; pp. 4:1–4:14



Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

length of the *geodesic* $\Pi(p, q)$, that is, the shortest path connecting p and q that is completely contained in P. We refer to this distance as the *geodesic distance* $\pi(p, q)$.

Related work. It is well known that if we have only a fixed set S of n sites, we can answer nearest neighbor queries efficiently by computing the Voronoi diagram of S and preprocessing it for planar point location. This requires $O(n \log n)$ preprocessing time, the resulting data structure uses linear space, and we can answer queries in $O(\log n)$ time. Voronoi diagrams have also been studied in case the set of sites is restricted to lie in a simple polygon P, and we measure the distance between two points p and q by their geodesic distance $\pi(p, q)$ [4, 18, 24, 25]. The approach of Hershberger and Suri [18] computes the geodesic Voronoi diagram in $O((m + n) \log(m + n))$ time, where m is the total number of vertices in the polygon P, and is applicable even if P has holes. Very recently, Oh and Ahn [24] presented an $O(m + n \log n \log^2 m)$ time algorithm. When $n \leq m/\log^3 m$ this improves the previous results. All these approaches allow for $O(\log(n + m))$ time nearest neighbor queries. However, they are efficient only when the set of sites S is fixed, as inserting or deleting even a single site may cause a linear number of changes in the Voronoi diagram.

To support nearest neighbor queries, it is, however, not necessary to explicitly maintain the (geodesic) Voronoi diagram. Bentley and Saxe [6] show that nearest neighbor searching is a *decomposable search problem*. That is, we can find the answer to a query by splitting S into groups, computing the solution for each group individually, and taking the solution that is best over all groups. This observation has been used in several other approaches for nearest neighbor searching with the Euclidean distance [1, 8, 11]. However, even with this observation, it is hard to get both polylogarithmic update and query time. Chan [8] was the first to achieve this. His data structure can answer Euclidean nearest neighbor queries in $O(\log^2 n)$ time, and supports insertions and deletions in $O(\log^3 n)$ and $O(\log^6 n)$ amortized time, respectively. Recently, Kaplan et al. [19] extended the result of Chan to more general, constant complexity, distance functions.

Unfortunately, the above results do not directly lead to an efficient solution to our problem. The function describing the geodesic distance may have complexity $\Theta(m)$, and thus the results of Kaplan et al. [19] do not apply. Moreover, even directly combining the decomposable search problem approach with the static geodesic Voronoi diagrams described above does not lead to an efficient solution, since every update incurs an $\Omega(m)$ cost corresponding to the complexity of the polygon. Only the very recent algorithm of Oh and Ahn [24] can be made amendable to such an approach. This results in an $O(n + m)$ size data structure with $O(\sqrt{n}(\log n + \log m))$ query time and $O(\sqrt{n} \log n \log^2 m)$ updates. Independently from Oh and Ahn, we developed a different data structure yielding similar results [3]. The core idea in both data structures is to represent the Voronoi diagram implicitly. Moreover, both approaches use similar primitives. In this manuscript, we build on the ideas from our earlier work, and significantly extend them to achieve polylogarithmic update and query times.

Our results. We develop a fully dynamic data structure to support nearest neighbor queries for a set of sites S inside a (static) simple polygon P. Our data structure allows us to locate the site in S closest to a query point $q \in P$, to insert a new site s into S, and to delete a site from S. Our data structure supports queries in $O(\log^2 n \log^2 m)$ time, insertions in $O(\log^5 n \log m + \log^4 n \log^3 m)$ amortized expected time, and deletions in $O(\log^7 n \log m + \log^6 n \log^3 m)$ amortized expected time. The space usage is $O(n \log^3 n \log m + m)$.

Furthermore, we show that using a subset of the tools and techniques that we develop, we can build an improved data structure for when there are no deletions. In this insertion-

only setting, queries take worst-case $O(\log^2 n \log^2 m)$ time, and insertions take amortized $O(\log n \log^3 m)$ time. We can also achieve these running times in case there are both insertions and deletions, but the order of these operations is known in advance. The space usage of this version is $O(n \log n \log m + m)$.

2 An overview of the approach

As in previous work on geodesic Voronoi diagrams [4, 25], we assume that P and S are in general position. That is, (i) no two sites s and t in S (ever) have the same geodesic distance to a vertex of P, and (ii) no three points (either sites or vertices) are colinear. Note that (i) implies that no bisector b_{st} between sites s and t contains a vertex of P.

Throughout the paper we will assume that the input polygon P has been preprocessed for two-point shortest path queries using the data structure by Guibas and Hershberger [13] (see also the follow up note of Hershberger [17]). This takes $O(m)$ time and allows us to compute the geodesic distance $\pi(p, q)$ between any pair of query points $p, q \in P$ in $O(\log m)$ time.

Dynamic Euclidean nearest neighbor searching. We briefly review the data structures for dynamic Euclidean nearest neighbor searching of Chan [8] and Kaplan et al. [19], and the concepts they use, as we will build on these results.

Let F be a set of bivariate functions, and let $\mathcal{A}(F)$ denote the arrangement of the (graphs of the) functions in $\mathbb{R}^3$. In the remainder of the paper we will no longer distinguish between a function and its graph. A point $q \in \mathbb{R}^3$ has *level* k if the number of functions in F that pass strictly below q is k. The *at most k-level* $L_{\leq k}(F)$ is the set of all points in $\mathbb{R}^3$ for which the level is at most k, and the k-level $L_k(F)$ is the boundary of that region.

Consider a collection X of points in $\mathbb{R}^3$ (e.g. a line segment), and let F_X denote the set of functions from F intersecting X. We refer to F_X as the *conflict list* of X. Furthermore, let $\underline{X}$ denote the vertical (downward) projection of X onto the x, y-plane.

A *pseudo-prism* is a constant complexity region in $\mathbb{R}^3$ that is bounded from above by a function, unbounded from below, and whose sides are surfaces vertical with respect to the z-direction. A *a k-shallow $(1/r)$-cutting* $\Lambda_{k,r}(F)$ is a collection of such pseudo-prisms with pairwise disjoint interiors whose union covers $L_{\leq k}(F)$ and for which each pseudo-prism intersects at most n/r functions in F [23]. Hence, for each region (pseudo-prism) $\nabla \in \Lambda_{k,r}(F)$, the conflict list F_∇ contains n/r functions. The number of regions in $\Lambda_{k,r}(F)$ is the *size* of the cutting. Matoušek [23] originally defined a shallow cutting in terms of simplicies, however using pseudo-prisms is more convenient in our setting. In this case the parameter r cannot become arbitrarily large, however we are mostly interested in k-shallow $O(k/n)$-cuttings. In the remainder of the paper we simply refer to such a cutting $\Lambda_k(F)$ as *a k-shallow cutting*. Observe that each pseudo-prism in $\Lambda_k(F)$ is intersected by $O(k)$ functions.

Let $f_s(x)$ be the distance from x to s. The data structures of Kaplan et al. [19] and Chan [8] actually maintain the *lower envelope* $L_0(F)$ of the set of distance functions $F = \{f_s \mid s \in S\}$. To find the site s closest to a query point q they can simply query the data structure to find the function f_s that realizes $L_0(F)$ at q. The critical ingredient in both data structures, as well as our own data structure, is an efficient algorithm to construct a shallow cutting of the functions in F. Chan and Tsakalidis [9] show that if F is a set of linear functions (i.e. planes in $\mathbb{R}^3$), we can compute a k-shallow cutting $\Lambda_k(F)$ of size $O(n/k)$ in $O(n \log n)$ time. Kaplan et al. [19] show how to compute a k-shallow cutting of size $O(n \log^2 n/k)$, in time $O(n \operatorname{polylog} n)$ for a certain class of constant complexity algebraic functions F. Since the geodesic distance function $f_s(x) = \pi(s, x)$ of a single site s may already have complexity

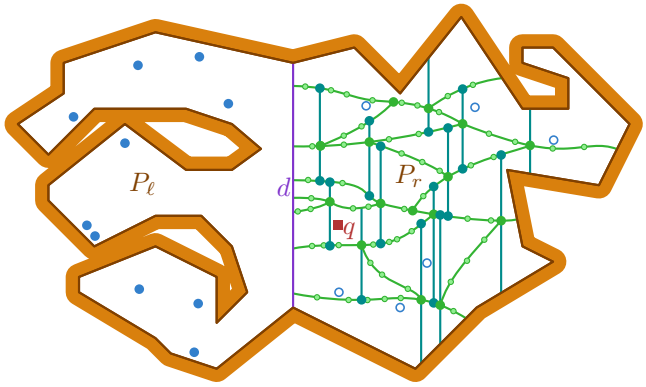


Figure 1 A schematic drawing of the downward projection of an implicit k-shallow cutting $\underline{\Lambda_k(F_\ell)}$ in P_r. The faces are pseudo-trapezoids. Neighboring pseudo-trapezoids share a vertical segment, or part of a bisector. We store only the degree one and three vertices (fat), and their topology.

$\Theta(m)$, any k-shallow cutting of such functions may have size $\Omega(m)$. To circumvent this issue, we allow the regions (pseudo-prisms) to have non-constant complexity. We do this in such a way that we can compactly represent each pseudo-prism, while still retaining some nice properties such as efficiently testing if a point lies inside it. Hence, in the remainder of the paper we will drop the requirement that the regions in a cutting need to have constant complexity.

The general approach. The general idea in our approach is to recursively partition the polygon into two roughly equal size sub-polygons P_ℓ and P_r that are separated by a diagonal. For the sites S_ℓ in the "left" subpolygon P_ℓ, we then consider their geodesic distance functions $F_\ell = \{f_s \mid s \in S_\ell\}$ restricted to the "right" subpolygon P_r, that is, $f_s(x) = \pi(s, x)$, for $x \in P_r$. The crucial part, and our main contribution, is that for these functions F_ℓ we can represent a vertical shallow cutting implicitly. See Fig. 1 for a schematic illustration. More specifically, in $O((n/k)\log^3 n(\log n + \log^2 m) + n\log^2 m + n\log^3 n \log m)$ expected time, we can build a representation of the shallow cutting of size $O((n/k)\log^2 n)$. We can then use this algorithm for building implicitly represented shallow cuttings in the data structure of Chan [8] and Kaplan et al. [19]. That is, we build and maintain the lower envelope $L_0(F_\ell)$. Symmetrically, for the sites in P_r, we maintain the lower envelope $L_0(F_r)$ that their distance functions F_r induce in P_ℓ. When we get a query point $q \in P_r$, we use $L_0(F_\ell)$ to find the site in P_ℓ closest to q in $O(\log^2 n \log m)$ time. To find the site in P_r closest to q, we recursively query in sub-polygon P_r. In total we query in $O(\log m)$ levels, leading to an $O(\log^2 n \log^2 m)$ query time. When we add or remove a site s we, insert or remove its distance function in $O(\log m)$ lower envelope data structures (one at every level). Every insertion takes $O(\log^5 n + \log^4 n \log^2 m)$ amortized expected time, and every deletion takes $O(\log^7 n + \log^6 \log^2 m)$ amortized expected time. Since every site is stored $O(\log m)$ times, this leads to the following main result.

▶ **Theorem 1.** *Let P be a simple polygon P with m vertices. There is a fully dynamic data structure of size $O(n\log^3 n \log m + m)$ that maintains a set of n point sites in P and allows for geodesic nearest neighbor queries in worst case $O(\log^2 n \log^2 m)$ time. Inserting a site takes $O(\log^5 n \log m + \log^4 n \log^3 m)$ amortized expected time, and deleting a site takes $O(\log^7 n \log m + \log^6 n \log^3 m)$ amortized expected time.*

The main complexity is in developing our implicit representation of the k-shallow cutting, and the algorithm to construct such a cutting. Once we have this algorithm we can directly plug it in into the data structure of Chan [8] and Kaplan et al. [19]. Our global strategy is similar to that of Kaplan et al. [19]: we compute an approximate k-level of $\mathcal{A}(F)$ –in our case an implicit representation of this approximate k-level– and then argue that, under certain conditions, this approximate k-level is actually a k-shallow cutting $\Lambda_k(F)$ of $\mathcal{A}(F)$. Our approximate k-level will be a t-level, for some appropriate t, on a random sample of the functions in F. So, that leaves us two problems: *i*) computing an implicit representation of a t-level, and *ii*) computing the conflict lists for all pseudo-prism in our cutting $\Lambda_k(F)$.

For problem *i*), representing the t-level implicitly, we use the connection between the t-level and the t^{th}-order Voronoi diagram. In Section 3 we describe a small, implicit representation of the t^{th}-order Voronoi diagram that still allows us to answer point location queries efficiently. Initially, we use the recent algorithm of Oh and Ahn [24] to construct this representation. In Section 5 we then design an improved algorithm for our particular use case.

For problem *ii*), computing the conflict lists, we will use a data structure developed by Chan [7] together with the implicit Voronoi diagrams that we developed in Section 5. We describe these results in more detail in Section 6. In Section 7, we then show in detail how we can combine all the different parts into a fully dynamic data structure for nearest neighbor queries. Omitted details are in the full version of this paper [2].

3 Implicit representations

Let $F = \{f_s \mid s \in S\}$ denote the set of geodesic distance functions inside the entire polygon P. Our implicit representation of a k-shallow cutting $\Lambda_k(F)$ is based on an implicit representation of the k-level in $\mathcal{A}(F)$. To this end, we first study higher order geodesic Voronoi diagrams.

Higher order Voronoi diagrams. Consider a set of n sites S, a domain $\mathcal{D}$, and a subset $H \subseteq S$ of size k. The *k^{th}-order Voronoi region* $V_k(H, S)$ is the region in $\mathcal{D}$ in which the points are closer to (a site in) H, with respect to some distance metric, than to any other subset $H' \subseteq S$ of size k. The *k^{th}-order Voronoi diagram* $\mathcal{V}_k(S)$ is the partition of $\mathcal{D}$ into such maximal regions $V_k(H, S)$ over all subsets $H \subseteq S$ of size k [21]. Liu et al. [22] study the *geodesic* k^{th}-order Voronoi diagram. They show that $\mathcal{V}_k(S)$ has complexity $O(k(n-k)+km)$. In particular, it consists of $O(k(n-k))$ degree one and degree three vertices, and $O(km)$ degree two vertices (and by our general position assumption, there are no vertices of degree more than three).

Consider a Voronoi region $V_k(H, S)$. Let $e_1, .., e_\ell$, be the edges bounding $\partial V_k(H, S)$, let H_j be the set of sites defining the other Voronoi region incident to e_j, and observe that $H_j \setminus H$ contains a single site q_j [21]. Let $Q = \{q_j \mid j \in [1, \ell]\}$ be the set of sites *neighboring* $V_k(H, S)$. Observe that these results imply that two adjacent regions $V_k(H, S)$ and $V_k(H_j, S)$ in $\mathcal{V}_k(S)$ are separated by a part of a bisector b_{st}, where $s = H \setminus H_j$ and $t = q_j$.

By combining the above two observations we can represent $\mathcal{V}_k(S)$ implicitly. That is, we store only the locations of these degree one and degree three vertices, the adjacency relations between the regions, and the pair of labels (s, t) corresponding to each pair (R_s, R_t) of neighboring regions. See also the recent result of Oh and Ahn [24]. It follows that the size of this representation is linear in the number of degree one and degree three vertices of $\mathcal{V}_k(S)$. We refer to this as the *topological complexity* of $\mathcal{V}_k(S)$.

Representing the k-level. Consider the partition of P into maximally connected regions in which all points in a region have the same k^{th} nearest site in S. Observe that this partition corresponds to the downward projection $\underline{L_k(F)}$ of the k-level $L_k(F)$. As we argue next, this partition is closely related to the k^{th}-order Voronoi diagram defined above.

Lee observes that there is a relation between the i^{th}-order Voronoi diagram and the $(i+1)^{\text{th}}$-order Voronoi diagram [21]. In particular, he shows that we can partition each i^{th}-order Voronoi region $V_i(H,S)$ into $(i+1)^{\text{th}}$ order Voronoi regions by intersecting $V_i(H,S)$ with the (first order) Voronoi diagram of the set of sites neighboring $V_i(H,S)$. More specifically:

▶ **Observation 2.** *Let $V_i(H,S)$ be a geodesic i^{th}-order Voronoi region, and let Q be the sites neighboring $V_i(H,S)$. For any point $p \in V_i(H,S)$, the $(i+1)$-closest site from p is the site $s \in Q$ for which $p \in \mathcal{V}(s,Q)$.*

▶ **Lemma 3.** *The topological complexity of $\underline{L_k(F)}$ is $O(k(n-k))$.*

Proof. By Observation 2, we can obtain $\underline{L_k(F)}$ from $\mathcal{V}_{k-1}(S)$ by partitioning every region $V_{k-1}(H,S)$ using the Voronoi diagram of the sites Q neighboring $V_{k-1}(H,S)$, and merging regions that have the same k^{th} nearest neighbor in the resulting subdivision. Thus, the vertices in $\underline{L_k(F)}$ appear either in $V_{k-1}(H,S)$ or in one of the newly created Voronoi diagrams.

There are $O((k-1)(n-k+1)) = O(k(n-k))$ degree one and degree three vertices that are also vertices in $V_{k-1}(H,S)$. Since the (first order) geodesic Voronoi diagram has $O(k)$ degree one and degree three vertices, a region $V_{k-1}(H,S)$ creates $|Q|$ new degree one and three vertices. Summing over all faces in $\mathcal{V}_{k-1}(H,S)$ this then sums to $O(k(n-k))$. ◀

As with $\mathcal{V}_k(S)$ we can represent $\underline{L_k(F)}$ implicitly by storing only the locations of the degree one and three vertices and the topology of the diagram. Note that if we can efficiently locate the region R_s of $\underline{L_k(F)}$ that contains a query point q, we can also easily compute the z-coordinate of $L_k(F)$ at q, simply by computing the geodesic distance $\pi(s,q)$. Since we preprocessed P for two-point shortest path queries this takes only $O(\log m)$ time (in addition to locating the region of $\underline{L_k(F)}$ containing q).

Representing a vertical decomposition. For every degree three and degree one vertex in $\underline{L_k(F)}$ we now extend a vertical segment up and down until it hits another edge of $\underline{L_k(F)}$. Observe that the topological complexity of the resulting *implicit vertical decomposition* $\underline{L_k^{\nabla}(F)}$ that we obtain still has topological complexity $O(k(n-k))$. Furthermore, each region in $\underline{L_k^{\nabla}(F)}$ is a *pseudo-trapezoid* $\underline{\nabla}$ that is bounded on the left and right either by a vertical segment or a piece of polygon boundary, and on the top and bottom by pieces of bisectors or a piece of polygon boundary. We refer to the four degree one or degree three vertices on the boundary of $\underline{\nabla}$ as the *corners* of $\underline{\nabla}$. See Fig. 2 for an illustration. In the remainder of the paper, we will no longer distinguish between $\underline{L_k^{\nabla}(F)}$ and its implicit representation. In Section 6, we will use such an implicit vertical decomposition to obtain an implicit representation of a shallow cutting.

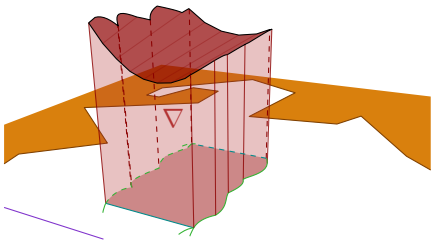


Figure 2 A pseudo prism ∇ and its projection $\underline{\nabla}$ (darker red) onto P_r.

Computing implicit representations. We can use the algorithm of Oh and Ahn [24] to compute the implicit representation of $\mathcal{V}_k(S)$ in $O(k^2 n \log n \log^2 m)$ time. To compute (the representation of) $\underline{L_k(F)}$ we first construct $\mathcal{V}_{k-1}(S)$, and for each region $V_{k-1}(H,S)$ in $\mathcal{V}_{k-1}(S)$, we again use their algorithm to compute the Voronoi diagrams $\mathcal{V}(Q)$ of the set neighbors Q. We then clip these diagrams to $V_{k-1}(H,S)$. This clipping can be done by a

breadth first search in $\mathcal{V}(Q)$, starting with one of the vertices that is also in $\partial V_{k-1}(H, S)$. This takes $O(|Q| \log |Q| \log^2 m)$ time in total. Summing over all faces in $\mathcal{V}_{k-1}(S)$ gives us again a running time of $O(k^2 n \log n \log^2 m)$. Finally, to compute $\underline{L_k^{\nabla}(F)}$ we need to insert two vertical extension segments at each vertex of $\underline{L_k(F)}$. We can find the other endpoint of each extension segment using a point location query. Thus, we obtain the following result.

▶ **Lemma 4.** *An implicit representation $\underline{L_k^{\nabla}(F)}$ of the k-level $L_k(F)$ that uses $O(k(n-k))$ space, can be computed in $O(k^2 n \log n \log^2 m)$ time. Using this representation, the pseudo-prism containing a query point (if it exists) can be determined in $O(\log n + \log m)$ time.*

In Section 5 we will show that if the sites defining the functions in F lie in one half of the polygon and we restrict the functions to the other half we can improve these results. Moreover, we can then compute an implicit representation of a k-shallow cutting of F.

4 Approximating the k-level

An x, y-monotone surface Γ is an *ε-approximation* of $L_k(F)$ if and only if Γ lies between $L_k(F)$ and $L_{(1+\varepsilon)k}(F)$. Following the same idea as in Kaplan et al. [19] we construct an ε-approximation of $L_k(F)$ as follows. We choose a random sample R of F of size $r = (cn/k\varepsilon^2) \log n$ and consider the t-level of $\mathcal{A}(R)$ for some randomly chosen level t in the range $[(1+\varepsilon/3)h, (1+\varepsilon/2)h]$. Here c and c' are some constants, $h = c'/\varepsilon^2 \log n$, and $\varepsilon \in [0, 1/2]$ is the desired approximation ratio. We now argue that $L_t(R)$ is an ε-approximation of $L_k(F)$.

Consider the range space $\mathcal{S} = (F, \mathcal{R})$, where each range in $\mathcal{R}$ is the subset of functions of F intersected by a downward vertical ray in the $(-z)$-direction. See Har-Peled [15] for details on range spaces. An important concept is the *VC-dimension* of $\mathcal{S}$, defined as the size of the largest subset $F' \subseteq F$ for which the number of sets in $\{F' \cap F_\rho \mid F_\rho \in \mathcal{R}\}$ is $2^{|F'|}$.

▶ **Lemma 5** (Lemma 2.3.5 of Aronov et al. [5])**.** *Let s, t, and u be three sites in P. Their bisectors b_{st} and b_{tu} intersect in at most a single point.*

▶ **Lemma 6.** *The VC-dimension of the range space $\mathcal{S}$ is finite.*

Proof. The range space $(F, \mathcal{R})$ is equivalent to $(S_\ell, \mathcal{D})$ where $\mathcal{D} \subseteq 2^{S_\ell}$ is the family of subsets of S_ℓ that lie within some geodesic distance of some point $p \in P$. That is $\mathcal{D} = \{\{s \in S_\ell \mid \pi(p, s) \leq z\} \mid p \in P, z \geq 0\}$. We now observe that any three points s, t, and u, define at most one geodesic disk (Lemma 5). Namely, the disk centered at the intersection point $p = b_{st} \cap b_{tu}$ and radius $\pi(p, s) = \pi(p, t) = \pi(p, u)$. This means the same argument used by as Har-Peled [15, Lemma 5.15] now gives us that the shattering dimension of $\mathcal{S}$ (see [15]) is constant (three). It then follows that the VC-dimension of $\mathcal{S}$ is finite. ◀

Since $\mathcal{S}$ has finite VC-dimension (Lemma 6), and R has size $r = (cn/k\varepsilon^2) \log n \geq \left(\frac{c}{\varepsilon^2 p}\left(\log \frac{1}{p} + \log \frac{1}{q}\right)\right)$, for $p = \frac{k}{2n}$ and $q = 1/n^b$ for some sufficiently large constant b, it follows that with high probability, R is a *relative* $(p, \frac{\varepsilon}{3})$*-approximation* for $\mathcal{S} = (F, \mathcal{R})$ [16]. So, for every range $H \in \mathcal{R}$, we have (whp.) that

$$\left|\frac{|H|}{|F|} - \frac{|H \cap R|}{|R|}\right| \leq \begin{cases} \frac{\varepsilon}{3}\frac{|H|}{|F|}, & \text{if } |H| \geq p|F|, \text{ and} \\ \frac{\varepsilon}{3}p & \text{if } |H| < p|F|. \end{cases} \tag{1}$$

Using exactly the same argument as Kaplan et al. [19] we then obtain the following result.

▶ **Lemma 7.** *The level $L_t(R)$ is an ε-approximation of the k-level $L_k(F)$.*

What remains is to show that the (expected) topological complexity of the t-level $L_t(R)$ is small.

▶ **Lemma 8.** *The expected topological complexity of level $L_t(R)$ is $O((n/k\varepsilon^5)\log^2 n)$.*

Proof. The lower envelope $L_0(R)$ of R has topological complexity $O(r)$, so by Clarkson and Shor [10] the total topological complexity of all levels in the range $[(1+\varepsilon/3)h, (1+\varepsilon/2)h]$ is $O(rh^2)$. Hence, the expected topological complexity of a level $L_t(R)$, with t randomly chosen from this range, is $O(rh^2/h\varepsilon) = O(rh/\varepsilon)$. Substituting $r = (cn/k\varepsilon^2)\log n$ and $h = c'/\varepsilon^2 \log n$, we get $O(nk/\varepsilon^5 \log^2 n)$ as claimed. ◀

Again as in Kaplan et al. [19], if $k < (1/\varepsilon^2)\log n$, we can skip the sampling of the set R, and directly take a random level t in $\mathcal{A}(F)$ in the range $[k, k(1+\varepsilon)]$. This level has also an expected topological complexity of at most $O((n/k\varepsilon^5)\log^2 n)$. Using Lemma 4 (and restarting the computation if the size of the cutting exceeds $O((n/k\varepsilon^5)\log^2 n)$) we then get:

▶ **Lemma 9.** *An ε-approximation of the k-level of $\mathcal{A}(F)$ that has topological complexity $O((n/k\varepsilon^5)\log^2 n)$ can be computed in expected $O((n/k\varepsilon^6)\log^3 n \log^2 m)$ time.*

The main difference between our approach and that of Kaplan et al. [19] is the range space used. In our approach, the ranges are defined by downward vertical rays, whereas in Kaplan et al. the ranges are defined by more general objects. For example, their range space includes a range consisting of the functions intersected by some other constant complexity algebraic function. This allows them to directly turn their approximate level into a shallow cutting since the boundaries of the pseudo-prisms correspond to ranges. Unfortunately, this idea does not directly extend to the geodesic setting, as the VC-dimension of such a range space may again depend on the complexity of the polygon. Therefore, we will use a different approach in Section 6.

5 Computing implicit representations in subpolygon P_r

Consider a diagonal d that splits the polygon P into two subpolygons P_ℓ and P_r, and assume without loss of generality that d is vertical and that P_r lies right of d. We consider only the sites S_ℓ in P_ℓ, and we restrict their functions $F = \{f_s \cap (P_r \times \mathbb{R}) \mid s \in S_\ell\}$ to P_r. We now present a more efficient algorithm to compute the implicit representation of $L_k(F)$ in this setting. To this end, we show that the two-point shortest path query data structure of Guibas and Hershberger [13] essentially gives us an efficient way of accessing the bisector b_{st} between a pair of sites without explicitly computing it. See the full version [2]. We use this to compute an implicit representation of the Voronoi diagram of S_ℓ in P_r. Building on these results, we can compute an implicit representation of the k^{th}-order Voronoi diagram $\mathcal{V}_k(S_\ell)$ in P_r, the k-level $L_k(F)$ of F in $P_r \times \mathbb{R}$, and finally an implicit vertical decomposition $\underline{L_k^\nabla}$ of $\underline{L_k(F)}$ in P_r. We sketch some of our results here. Refer to the full version [2] for the details.

Computing an implicit Voronoi diagram. Our main idea for constructing $\mathcal{V} = \mathcal{V}(S_\ell)$ in P_r, is that we can consider it as a *Hamiltonian abstract Voronoi diagram*. Such a Voronoi diagram can be constructed in $O(Xn)$ time [20], where X is the time required for certain geometric primitives. We can implement these primitives to run in $O(\log^2 m)$ time, resulting in a $O(n \log^2 m)$ time algorithm to compute $\mathcal{V}$.

A Voronoi diagram is *Hamiltonian* if there is a curve –in our case the diagonal d– that intersects all regions exactly once, and furthermore this holds for all subsets of the sites [20]. Let T_ℓ be the subset of sites from S_ℓ whose Voronoi regions intersect d, and thus occur in $\mathcal{V}$.

▶ **Lemma 10.** *The Voronoi diagram $\mathcal{V}(T_\ell)$ in P_r is a Hamiltonian abstract Voronoi diagram.*

To construct $\mathcal{V} = \mathcal{V}(S_\ell) = \mathcal{V}(T_\ell)$ we need the set of sites T_ℓ whose Voronoi regions intersect d, and the order in which they do so. The following lemma is the key insight that allows us to extract T_ℓ from S_ℓ in $O(n \log^2 m)$, assuming that we maintain the sites in S_ℓ in a balanced binary search tree ordered on increasing distance from the bottom endpoint of d.

▶ **Lemma 11.** *Let $s_1, .., s_n$ denote the sites in S_ℓ ordered by increasing distance from the bottom-endpoint p of d, and let $t_1, .., t_z$ be the subset $T_\ell \subseteq S_\ell$ of sites whose Voronoi regions intersect d, ordered along d from bottom to top. For any pair of sites $t_a = s_i$ and $t_c = s_j$, with $a < c$, we have that $i < j$.*

Once we have the set of sites T_ℓ, we construct $\mathcal{V}$ in $O(n \log^2 m)$ time, using the algorithm of Klein and Lingas [20]. Refer to the full version [2] for the details.

▶ **Lemma 12.** *For $s, t \in S_\ell$, the part of the bisector $b^*_{st} = b_{st} \cap P_r$ that lies in P_r is x-monotone.*

It follows from Lemma 12 that we can use the approach of Edelsbrunner, Guibas, and Stolfi [12] to preprocess $\mathcal{V}$ for planar point location queries, and obtain the following result.

▶ **Lemma 13.** *Given a set of n sites S_ℓ in P_ℓ, ordered by increasing distance from the bottom-endpoint of d, the forest $\mathcal{V}$ representing the Voronoi diagram of S_ℓ in P_r can be computed in $O(n \log^2 m)$ time. Given $\mathcal{V}$, finding the site $s \in S_\ell$ closest to a query point $q \in P_r$ requires $O(\log n \log m)$ time.*

Computing $\mathcal{V}_k(S_\ell)$, $L_k(F)$, and an implicit vertical decomposition of the k-level. By combining the algorithm from Lemma 13 with the iterative approach of Lee [21] we can construct an implicit representation of the k^{th}-order Voronoi diagram $\mathcal{V}_k(S_\ell)$ in P_r, or similarly the k-level of F. We then decompose the downward projection $\underline{L_k(F)}$ into *pseudo*-trapezoids, giving us an implicit vertical decomposition.

▶ **Lemma 14.** *A representation $\underline{L^\nabla_k}$ of the k-level $L_k(F)$ consisting of $O(k(n-k))$ pseudo-trapezoids can be computed in $O(k^2 n(\log n + \log^2 m))$ time. Given a query point $q \in P_r$, the k-nearest site in S_ℓ can be reported in $O(\log n \log m)$ time.*

6 An implicit shallow cutting of the geodesic distance function

Let F again denote the set of geodesic distance functions that the sites S_ℓ in P_ℓ induce in P_r. We now argue that we can compute an implicit k-shallow cutting $\Lambda_k(F)$ for these functions.

As in Section 4, let R be our random sample of size r, and let $L_t(R)$ be our approximate k-level of $\mathcal{A}(F)$. Let $\underline{L^\nabla_t(R)}$ be the vertical decomposition of $L_t(R)$. We now raise every pseudo-trapezoid in $\underline{L^\nabla_t(R)}$ to the t-level. Denote the result by Λ. Let $F_p = F_{\rho(p)}$ denote the conflict list of $p \in \mathbb{R}^3$, i.e., the functions intersecting the vertical downward half-line $\rho(p)$ starting in p.

▶ **Lemma 15.** *Let ∇ be a pseudo prism in Λ. The conflict list F_∇ of ∇ is the union of the conflict lists of its corners W, i.e. $F_\nabla = \bigcup_{v \in W} F_v$.*

Proof. Let f_s be the function defining the ceiling of ∇. We have that $F' = \bigcup_{v \in W} F_v \subseteq F_\nabla$ by definition, so we focus on proving $F_\nabla \subseteq F'$. Assume by contradiction that $f_t \in F_\nabla$, but

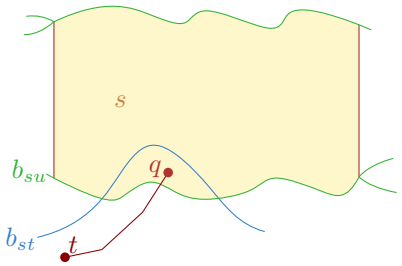


Figure 3 Since the bisectors restricted to P_r are x-monotone it follows that if a site t conflicts with a prism ∇, it must conflict with a corner of ∇.

$f_t \notin F'$. So, there is a point $q \in \underline{\nabla}$ for which $\pi(t,q) < \pi(s,q)$, but $\pi(s,v) < \pi(t,v)$ for all corners $v \in W$. Hence, all four corners lie on the "s-side" of b^*_{st}, whereas p lies on the "t-side" of b^*_{st}. Assume without loss of generality that s is closer to the points above b^*_{st} (and thus all corners lie above b^*_{st}). See Fig. 3. Since b^*_{st} is x-monotone (Lemma 12) it must intersect the bottom edge of $\underline{\nabla}$ twice. This bottom edge is part of a single bisector b^*_{su}, for some $f_u \in F$. However, by Lemma 5 b^*_{st} and b^*_{su} intersect at most once. Contradiction. ◀

▶ **Theorem 16.** *Λ is a vertical k-shallow $(k(1+\varepsilon)/n)$-cutting of $\mathcal{A}(F)$ whose topological complexity, and thus its size, is $O((n/k\varepsilon^5)\log^2 n)$. Each pseudo-prism in Λ intersects at least k and at most $4k(1+\varepsilon)$ functions in F.*

Proof. By Lemma 8 Λ consists of $O((n/k\varepsilon^5)\log^2 n)$ regions. Note that all regions are pseudo-prisms. Lemma 15 then gives us that the conflict list of each pseudo-prism is contained in the conflict lists of its at most four corners. ◀

6.1 Computing the conflict lists

Using Lemma 14 we can construct an implicit representation of the k-shallow cutting $\Lambda = \Lambda_k(F)$. So, all that remains is to compute the conflict lists of the pseudo-prisms. By Lemma 15 it is sufficient to compute the conflict lists of the four corner points of each pseudo-prism. Next, we show how to do this in $O(n(\log^3 n \log m + \log^2 m))$ expected time.

We use the same approach as used by Chan [7]. That is, we first build a data structure on our set of functions F so that for a vertical query line ℓ (in $\mathbb{R}^3$) and a value k, we can report the lowest k functions intersected by ℓ in $O((\log n + k)\log m)$ expected time. We then extend this to report only the functions that pass strictly below some point $q \in \mathbb{R}^3$. To compute the conflict lists of all corners in $\Lambda_k(F)$ we repeatedly query this data structure.

The data structure. Our data structure consists of a hierarchy of the lower envelopes of random samples $R_0 \subset R_1 \subset .. \subset R_{\log n}$, where $|R_i| = 2^i$. For each set R_i we store an implicit vertical decomposition representing the (the downward projection of the) lower envelope $L_{0,i} = L_0(R_i)$. This decomposes the space below $L_{0,i}$ into pseudo-prisms. For each such pseudo-prism ∇ we store its conflict list F_∇ with respect to F, i.e. the functions from (the entire set) F that intersect ∇. The following lemma shows that for each R_i, the expected amount of space used is $O(n)$. The total expected space used is thus $O(n \log n)$.

▶ **Lemma 17.** *Let $r \in [1,n]$ and consider a random sample R of F of size r. (i) The expected value of $\sum_\nabla |F_\nabla|$ over all pseudo-prisms below $L_0(R)$ is $O(n)$, and (ii) For any vertical*

line ℓ, the expected value of $|F_\nabla|$, where ∇ is the pseudo-prism of $L_0(R)$ intersected by ℓ, is $O(n/r)$.

Proof. The first statement follows directly from a Clarkson and Shor style sampling argument. More specifically, from what Har-Peled [15] calls the "Bounded moments theorem" (Theorem 8.8). The second statement then follows directly from the first statement. ◀

Building the data structure. For each set R_i, we use the algorithm from Section 5 to construct an implicit vertical decomposition of $\underline{L_{0,i}}$. To this end, we need to order the (sites corresponding to the) functions in R_i on increasing distance to the bottom endpoint of the diagonal d. For $R_{\log n} = F$ we do this in $O(n(\log n + \log m))$ time. For R_{i-1} we do this by filtering the ordered set R_i in linear time. Since the sizes of R_i are geometrically decreasing, it follows that we spend $O(n(\log n + \log^2 m))$ time in total.

▶ **Lemma 18.** *Let $f_s \in F \setminus R$ be a function that intersects a pseudo-prism of $L_0(R)$, let T be the set of sites whose functions contribute to $L_0(R)$, ordered on increasing distance from the bottom endpoint of d, and let t and u be the predecessor and successor of s in T, respectively. The vertex $v \in \underline{L_0(R)}$ that represents $d \cap b_{tu}$ is closer to s than to t and u.*

Proof. If f_s intersects a pseudo prism of $L_0(R)$ then there is a point $q \in P_r$ for which s is closer than all other sites in T. It follows that there must be a point on the diagonal d that is closer to s than to all other sites in T. Lemma 11 then gives us that the Voronoi region of s (with respect to $R \cup \{s\}$) on d must lie in between that of t and u (if these still contribute a Voronoi region). Therefore, t and u no longer have a vertex of $\mathcal{V}(R \cup \{s\})$ on d. Since t and u were the closest sites to v in R, this implies that v must lie in the Voronoi region of s, hence s is closer to v than t and u. ◀

By Lemma 18 we can now compute the conflict lists of the cells in $L_{0,i}$ as follows. For each function $f_s \in F \setminus R_i$ we find the vertex v defined in Lemma 18. If s is further from v than the sites defining it, then f_s does not conflict with any pseudo-prism in $L_{0,i}$. Otherwise, we find *all* (degree one or degree three) vertices of $\underline{L_{0,i}}$ that conflict with s. Since Voronoi regions are simply connected, we can do this using a breadth first search in $\underline{L_{0,i}}$, starting from vertex v. When we have this information for all functions in $F \setminus R_i$, we actually also know for every vertex v in $L_{0,i}$ which functions $F \setminus R_i$ pass below it. That is, we have the conflict lists for all vertices v. The conflict list of a pseudo-prism in $L_{0,i}$ is then simply the union of the conflict lists of its four corners (Lemma 15).

Given the ordering of all sites in S on increasing distance to the bottom endpoint of d, we can find the initial vertices for all functions in $F \setminus R_i$ in $O(|R_i| \log m))$ time. For every other reported conflict we spend $O(\log m)$ time, and thus computing the conflict lists for all cells in $L_{0,i}$ takes $O(\sum_{\nabla \in L_{0,i}} |F_\nabla| \log m)$ time. By Lemma 17 this sums to $O(n \log m)$ in expectation. Summing over all $O(\log n)$ random samples, it follows that we spend $O(n \log n \log m)$ expected time to compute all conflict lists. The total expected time to build the data structure is thus $O(n(\log^2 m + \log n \log m))$.

Querying. The query algorithm is exactly as in Chan [7]. The main idea is to use a query algorithm that may fail, depending on some parameter δ, and then query with varying values of δ until it succeeds. The query algorithm locates the cell ∇ in $L_0(R_i)$ stabbed by the vertical line ℓ, for $i = \lceil \log \lceil n\delta/k \rceil \rceil$. If $|F_\nabla| > k/\delta^2$ or $|F_\nabla \cap \ell| < k$ the query algorithm simply fails. Otherwise it reports the k lowest functions intersecting ℓ. Since computing the intersection of a function f_s with ℓ takes $O(\log m)$ time, the running time is $O((\log n + k/\delta^2) \log m)$. Using

three independent copies of the data structure, and querying with $\delta = 2^{-j}$ for increasing j gives us an algorithm that always succeeds in $O((\log n + k)\log m)$ time. Refer to Chan [7] for details. We can now also report all functions that pass below a point q by repeatedly querying with the vertical line through q and doubling the value of k. This leads to a query time of $O((\log n + k)\log m)$, where k is the number of functions passing below q.

▶ **Theorem 19.** *There is a data structure of size $O(n \log n)$ that allows reporting the k lowest functions in $\mathcal{A}(F)$ intersected by a vertical line through a query point $q \in P_r$, that is, the k-nearest neighbors of a query point q, or all k functions that pass below q, in $O((\log n + k)\log m)$ time. Building the data structure takes $O(n(\log n \log m + \log^2 m))$ expected time.*

Computing a shallow cutting. To construct a shallow cutting we now take a random sample R of size r, build an implicit representation of the t-level in this sample, and then construct the above data structure to compute the conflict lists. By Lemma 14 constructing the implicit representation of $L_t(R)$ takes $O(t^2 r(\log r + \log^2 m))$ time. Plugging in $r = (cn/k\varepsilon^2)\log n$, $t = \Theta(1/\varepsilon^2 \log n)$, and $\varepsilon = 1/2$, this takes $O((n/k)\log^3 n(\log n + \log^2 m))$ expected time.

Constructing the query data structure takes $O(n(\log n \log m + \log^2 m))$ time. We then query it with all degree three and degree one vertices in Λ. The total size of these conflict lists is $O(n \log^2 n)$ (Theorem 16). So, this takes $O(n \log^3 n \log m)$ time in total. We conclude:

▶ **Theorem 20.** *A k-shallow cutting $\Lambda_k(F)$ of F of topological complexity $O((n/k)\log^2 n)$ can be computed in $O((n/k)\log^3 n(\log n + \log^2 m) + n\log^2 m + n\log^3 n \log m)$ expected time.*

7 Putting everything together

Kaplan et al. [19] essentially prove the following result, which, combined with Theorem 20 gives us an efficient data structure to answer nearest neighbor queries when sites are in P_ℓ and the query points are in P_r.

▶ **Lemma 21** (Kaplan et al. [19])**.** *Given an algorithm to construct a k-shallow cutting Λ of size $S(n,k)$ on n functions in $T(n,k)$ time, and such that locating the cell ∇ in Λ containing a query point q takes $Q(n,k)$ time, we can construct a data structure of size $O(S(n,k)\log n)$ that maintains a dynamic set of at most n functions F and can report the function that realizes the lower envelope $L_0(F)$ at a query point q in $O(Q(n,1)\log n)$ time. Inserting a new function in F takes $O((T(n,1)/n)\log n)$ amortized time, and deleting a function from F takes $O((T(n,1)/n)\log^3 n)$ amortized time.*

Our main data structure is a balanced binary tree, corresponding to a balanced decomposition of P into sub-polygons [14], in which each node stores two copies of the data structure from Lemma 21. A node in the tree corresponds to a subpolygon P' of P, and a diagonal d that splits P' into two roughly equal size subpolygons P_ℓ and P_r. One copy of our data structure associated with this node stores the sites in S_ℓ and can answer queries in P_r. The other copy stores the sites in S_r and can answer queries in P_ℓ. Since the balanced hierarchical decomposition consists of $O(\log m)$ layers, every site is stored $O(\log m)$ times. This results in an $O(n \log^3 n \log m + m)$ size data structure. To answer a query q, we query $O(\log m)$ data structures, one at every level of the tree, and we report the site that is closest over all.

▶ **Theorem 1.** *Let P be a simple polygon P with m vertices. There is a fully dynamic data structure of size $O(n \log^3 n \log m + m)$ that maintains a set of n point sites in P and allows for geodesic nearest neighbor queries in worst case $O(\log^2 n \log^2 m)$ time. Inserting a site takes $O(\log^5 n \log m + \log^4 n \log^3 m)$ amortized expected time, and deleting a site takes $O(\log^7 n \log m + \log^6 n \log^3 m)$ amortized expected time.*

Proof. Theorem 20 yields $T(n,k) = O((n/k)\log^3 n(\log n+\log^2 m)+n\log^2 m+n\log^3 n\log m)$, $S(n,k) = O(n\log^2 n)$, and $Q(n,k) = O(\log n\log m)$, where m is the size of our polygon. Therefore, $T(n,1)/n = O(\log^4 n+\log^3 n\log^2 m)$. Plugging in these results in Lemma 21 and using that the balanced decomposition consists of $O(\log m)$ levels completes the proof. ◀

In case there are only insertions and no deletions, or the full sequence of updates is known in advance, we can design an alternative data structure, using a subset of our results. The main idea is still to recursively partition the polygon subpolygons P_ℓ and P_r. Instead of building a dynamic lower envelope data structure of the sites $S_\ell = S \cap P_\ell$ in P_r, we further split the sites S_ℓ into subsets $S_1, .., S_k$. For each subset we use the algorithm from Section 5 to build (an implicit representation of) the Voronoi diagram they induce in P_r. To answer a query (of a query point in P_r) we simply query *all* k (implicit) Voronoi diagrams. To update the data structure we simply rebuild the Voronoi diagram(s) of the affected subset(s). By appropriately splitting S_ℓ we get $O(\log^2 n\log^2 m)$ query time and $O(\log n\log^3 m)$ amortized update time, using $O(n\log n\log m+m)$ space. See the full version [2] for the details.

References

1. Pankaj K. Agarwal and Jiří Matoušek. Dynamic Half-Space Range Reporting and its Applications. *Algorithmica*, 13(4):325–345, 1995.
2. Pankaj Agarwal K., Lars Arge, and Frank Staals. Improved dynamic geodesic nearest neighbor searching in a simple polygon. *CoRR*, abs/1803.05765, 2018. URL: `http://arxiv.org/abs/1803.05765`.
3. Lars Arge and Frank Staals. Dynamic geodesic nearest neighbor searching in a simple polygon. *CoRR*, abs/1707.02961, 2017. URL: `http://arxiv.org/abs/1707.02961`.
4. Boris Aronov. On the Geodesic Voronoi Diagram of Point Sites in a Simple Polygon. *Algorithmica*, 4(1):109–140, 1989.
5. Boris Aronov, Steven Fortune, and Gordon Wilfong. The furthest-site geodesic voronoi diagram. *Discrete & Computational Geometry*, 9(3):217–255, Mar 1993.
6. Jon Louis Bentley and James B Saxe. Decomposable searching problems I. Static-to-dynamic transformation. *Journal of Algorithms*, 1(4):301–358, 1980.
7. Timothy M. Chan. Random Sampling, Halfspace Range Reporting, and Construction of ($\leq k$)-levels in Three Dimensions. *SIAM Journal on Computing*, 30(2):561–575, 2000.
8. Timothy M. Chan. A Dynamic Data Structure for 3-D Convex Hulls and 2-D Nearest Neighbor Queries. *Journal of the ACM*, 57(3):16:1–16:15, 2010.
9. Timothy M. Chan and Konstantinos Tsakalidis. Optimal Deterministic Algorithms for 2-d and 3-d Shallow Cuttings. In *Proc. 31st International Symposium on Computational Geometry*, volume 34 of *Leibniz International Proceedings in Informatics*, pages 719–732. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015.
10. Kenneth L. Clarkson and Peter W. Shor. Applications of random sampling in computational geometry, II. *Discrete & Computational Geometry*, 4:387–421, 1989.
11. David Dobkin and Subhash Suri. Maintenance of Geometric Extrema. *Journal of the ACM*, 38(2):275–298, 1991.
12. Herbert Edelsbrunner, Leo J. Guibas, and Jorge Stolfi. Optimal Point Location in a Monotone Subdivision. *SIAM Journal on Computing*, 15(2):317–340, 1986.
13. 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.
14. Leonidas J. Guibas, John Hershberger, Daniel Leven, Micha Sharir, and Robert E. Tarjan. Linear-Time Algorithms for Visibility and Shortest Path Problems Inside Triangulated Simple Polygons. *Algorithmica*, 2(1):209–233, 1987.

15 Sariel Har-Peled. *Geometric Approximation Algorithms*, volume 173. American mathematical society Boston, 2011.

16 Sariel Har-Peled and Micha Sharir. Relative (p, ϵ)-approximations in Geometry. *Discrete & Computational Geometry*, 45(3):462–496, Apr 2011.

17 John Hershberger. A new data structure for shortest path queries in a simple polygon. *Information Processing Letters*, 38(5):231–235, 1991.

18 John Hershberger and Subhash Suri. An Optimal Algorithm for Euclidean Shortest Paths in the Plane. *SIAM Journal on Computing*, 28(6):2215–2256, 1999.

19 Haim Kaplan, Wolfgang Mulzer, Liam Roditty, Paul Seiferth, and Micha Sharir. Dynamic Planar Voronoi Diagrams for General Distance Functions and their Algorithmic Applications. In *Proc. 28th Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2017.

20 Rolf Klein and Andrzej Lingas. *Hamiltonian abstract Voronoi diagrams in linear time*, pages 11–19. Springer Berlin Heidelberg, Berlin, Heidelberg, 1994.

21 Der-Tsai Lee. On k-nearest neighbor voronoi diagrams in the plane. *IEEE Transactions on Computers*, C-31(6):478–487, June 1982.

22 Chih-Hung Liu and D. T. Lee. Higher-order geodesic voronoi diagrams in a polygonal domain with holes. In *Proc. 24th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1633–1645, 2013. doi:10.1137/1.9781611973105.117.

23 Jiří Matoušek. Reporting points in halfspaces. *Computational Geometry Theory and Applications*, 2(3):169–186, 1992.

24 Eunjin Oh and Hee-Kap Ahn. Voronoi Diagrams for a Moderate-Sized Point-Set in a Simple Polygon. In *Proc. 33rd International Symposium on Computational Geometry*, volume 77 of *Leibniz International Proceedings in Informatics*, pages 52:1–52:15. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017.

25 Evanthia Papadopoulou and Der-Tsai Lee. A New Approach for the Geodesic Voronoi Diagram of Points in a Simple Polygon and Other Restricted Polygonal Domains. *Algorithmica*, 20(4):319–352, 1998.

$\widetilde{O}(n^{1/3})$-Space Algorithm for the Grid Graph Reachability Problem

Ryo Ashida
Department of Mathematical and Computing Science, Tokyo Institute of Technology
(c/o Professor Osamu Watanabe)
Tokyo, Japan
ashida1@is.titech.ac.jp

Kotaro Nakagawa
JMA SYSTEMS Corporation
Tokyo, Japan
kootaroonakagawa@gmail.com

Abstract

The directed graph reachability problem takes as input an n-vertex directed graph $G = (V, E)$, and two distinguished vertices s and t. The problem is to determine whether there exists a path from s to t in G. This is a canonical complete problem for class NL. Asano et al. proposed an $\widetilde{O}(\sqrt{n})$ space[1] and polynomial time algorithm for the directed grid and planar graph reachability problem. The main result of this paper is to show that the directed graph reachability problem restricted to grid graphs can be solved in polynomial time using only $\widetilde{O}(n^{1/3})$ space.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases graph reachability, grid graph, graph algorithm, sublinear space algorithm

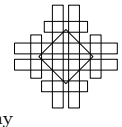
Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.5

Related Version A full version of this paper is available at `http://arxiv.org/abs/1803.07097`

1 Introduction

The graph reachability problem, for a graph $G = (V, E)$ and two distinct vertices $s, t \in V$, is to determine whether there exists a path from s to t. This problem characterizes many important complexity classes. The *directed graph reachability problem* is a canonical complete problem for the nondeterministic log-space class, NL. Reingold showed that the *undirected graph reachability problem* characterizes the deterministic log-space class, L[8]. As with P vs. NP problem, whether L=NL or not is a major open problem. This problem is equivalent to whether the directed graph reachability problem is solvable in deterministic log-space. There exist two fundamental solutions for the directed graph reachability problem, breadth first search, denoted as BFS, and Savitch's algorithm. BFS runs in $O(n)$ space and $O(m)$ time, where n and m are the number of vertices and edges, respectively. For Savitch's algorithm, we use only $O(\log^2 n)$ space but require $\Theta(n^{\log n})$ time. BFS needs short time but large space. Savitch's algorithm uses small space but super polynomial time. A natural question is whether we can make an efficient deterministic algorithm in both space and time for the directed graph reachability problem. In particular, Wigderson proposed a problem that does

[1] In this paper "$\widetilde{O}(s(n))$ space" means $O(s(n))$ words intuitively and precisely $O(s(n) \log n)$ space.



there exist an algorithm for the directed graph reachability problem that uses polynomial time and $O(n^{\varepsilon})$ space, for some $\varepsilon < 1$? [11], and this question is still open. The best known polynomial time algorithm, shown by Barns, Buss, Ruzzo and Schieber, uses $O(n/2^{\sqrt{\log n}})$ space [4].

For some restricted graph classes, better results are known. Stolee and Vinodchandran showed that for any $0 < \varepsilon < 1$, the reachability problem for directed acyclic graph with $O(n^{\varepsilon})$ sources and embedded on a surface with $O(n^{\varepsilon})$ genus can be solved in polynomial time and $O(n^{\varepsilon})$ space [9]. A natural and important restricted graph class is the class of planar graphs. The *planar graph reachability problem* is hard for L, and in the unambiguous log-space class, UL [5], which is a subclass of NL. Imai et al. gave an algorithm using $O(n^{1/2+\varepsilon})$ space and polynomial time for the planar graph reachability problem [2, 7]. Moreover Asano et al. devised a efficient way to control the recursion, and proposed a polynomial time and $\widetilde{O}(\sqrt{n})$ space algorithm for the planar graph reachability problem [3]. In this paper, we focus on the *grid graph reachability problem*, where grid graphs are special cases of planar graphs. Allender et al. showed the planar graph reachability problem is log-space reducible to the grid graph reachability problem [1]. By using the algorithm of Asano et al., we can solve the grid graph reachability problem in $\widetilde{O}(\sqrt{n})$ space and polynomial time. The main result of this paper is to show an $\widetilde{O}(n^{1/3})$ space and polynomial time algorithm for the directed grid graph reachability problem.

▶ **Theorem 1** ([3]). *There exists an algorithm that decides directed planar graph reachability in polynomial time and $\widetilde{O}(\sqrt{n})$ space. (We refer to this algorithm by* PlanarReach *in this paper.)*

2 Preliminaries and an outline of the algorithm

We will use the standard notions and notations for algorithms, complexity measures, and graphs without defining them. We consider mainly directed graphs, and a graph is assumed to be a directed graph unless it is specified as a undirected graph. Throughout this paper, for any set X, $|X|$ denotes the number of elements in X. We refer to the maximum and minimum elements of X as $\max X$ and $\min X$, respectively. Consider any directed graph $G = (V, E)$. For any $u, v \in V$, a directed edge e from u to v is denoted as $e = (u, v)$; on the other hand, the tail u and the head v of e are denoted as $t(e)$ and $h(e)$, respectively. For any $U \subseteq V$, let $G[U]$ denote the subgraph of G induced by U.

Recall that a grid graph is a graph whose vertices are located on grid points, and whose vertices are adjacent only to their immediate horizontal or vertical neighbors. We refer to a vertex on the boundary of a grid graph as a *rim vertex*. For any grid graph G, we denote the set of the rim vertices of G as R_G.

Computational model

For discussing sublinear-space algorithms formally, we use the standard multi-tape Turing machine model. A multi-tape Turing machine consists of a read-only input tape, a write-only output tape, and a constant number of work tapes. The space complexity of this Turing machine is measured by the total number of cells that can be used as its work tapes.

For the sake of explanation, we will follow a standard convention and give a sublinear-space algorithm by a sequence of constant number of sublinear-space subroutines $A_1, \ldots, A_k$ such that each A_i computes, from its given input, some output that is passed to A_{i+1} as an input. Note that some of these outputs cannot be stored in a sublinear-size work tape;

nevertheless, there is a standard way to design a sublinear-space algorithm based on these subroutines. The key idea is to compute intermediate inputs every time when they are necessary. For example, while computing A_i, when it is necessary to see the jth bit of the input to A_i, simply execute A_{i-1} (from the beginning) until it yields the desired jth bit on its work tape, and then resume the computation of A_i using this obtained bit. It is easy to see that this computation can be executed in sublinear-space. Furthermore, while a large amount of extra computation time is needed, we can show that the total running time can be polynomially bounded if all subroutines run in polynomial-time.

Outline of the algorithm

We show the outline of our algorithm. Our algorithm uses the algorithm PlanarReach for the planar graph reachability. We assume both $\sqrt{n}$ and $n^{1/3}$ are integers for simplicity. Let G be an input $\sqrt{n} \times \sqrt{n}$ grid graph with n vertices.

1. Separate G into $n^{1/3} \times n^{1/3}$ small grid graphs, or "blocks". There are $n^{1/3}$ blocks, and each block contains $n^{2/3}$ vertices.
2. Transform each block B into a special planar graph, "gadget graph", with $O(n^{1/3})$ vertices. The reachability among the vertices in R_B should be unchanged. The total number of vertices in all blocks becomes $O(n^{2/3})$.
3. We apply the algorithm PlanarReach to the transformed graph of size $O(n^{2/3})$, then the reachability is computable in $\widetilde{O}\left(\sqrt{n^{2/3}}\right) = \widetilde{O}(n^{1/3})$ space.

In step 1 and 2, we reduce the number of vertices in the graph G while keeping the reachability between the rim vertices of each block so that we can solve the reachability problem of the original graph. Then to this transformed graph we apply PlanarReach in step 3, which runs in $\widetilde{O}(n^{1/3})$ space.

▶ **Theorem 2.** *There exists an algorithm that computes the grid graph reachability in polynomial-time and $\widetilde{O}(n^{1/3})$ space.*

The start vertex s (resp., the end vertex t) may not be on the rim of any block. In such a situation, we make an additional block so that s (resp., t) would be on the rim of the block. This operation would not increase the time and space complexity. In this paper, we assume that s (resp., t) is on the rim of some block.

3 Graph transformation

In this section, we explain an algorithm that modifies each block and analyze time and space complexity of the algorithm. Throughout this section, we let a directed graph $G_0 = (V_0, E_0)$ denote a block of the input grid graph, and let V_0^{rim} denote the set of its rim vertices. We use N to denote the number of vertices of the input grid graph and n to denote $|V_0^{\mathrm{rim}}|$, which is $O(N^{1/3})$; note, on the other hand, that we have $|V_0| = O(n^2) = O(N^{2/3})$. Our task is to transform this G_0 to a plane "gadget graph", an augmented plane graph, $\widetilde{G}_{\mathsf{p}}$ with $O(n) = O(N^{1/3})$ vertices including V_0^{rim} so that the reachability among vertices in V_0^{rim} on G_0 remains the same on $\widetilde{G}_{\mathsf{p}}$.

There are two steps for this transformation. We first transform G_0 to a circle graph G_0^{cir}, and then obtain $\widetilde{G}_{\mathsf{p}}$ from the circle graph.

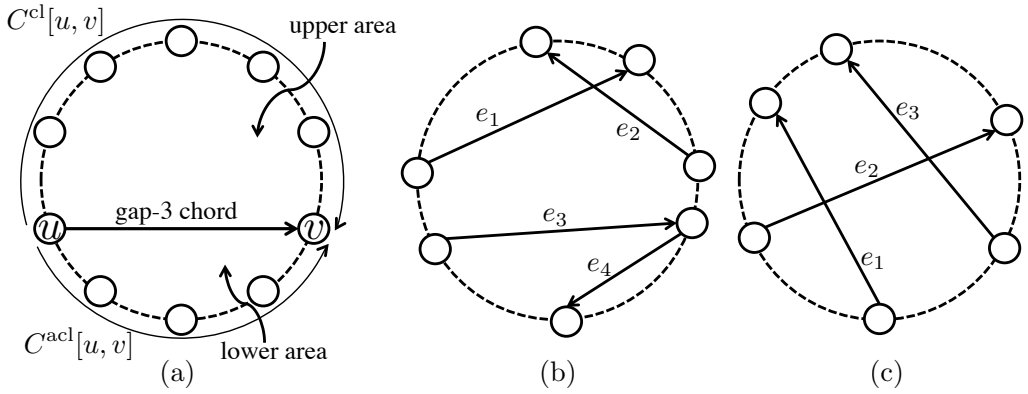


Figure 1 An example of the notions on chords. (a) a figure showing a chord, arcs, a lower area, an upper area, (b) a figure showing crossing chords (e_1 and e_2) and semi-crossing chords (e_3 and e_4) and (c) separating chords (e_3 separates e_1 and e_2).

3.1 Circle graph

We introduce the notion of "circle graph". A *circle graph* is a graph embedded on the plane so that all its vertices are placed on a cycle and all its edges are drawn inside of the cycle. Note that a circle graph may not have an edge between a pair of adjacent vertices on the cycle. We introduce some basic notions on circle graphs. Consider any circle graph $G = (V, E)$, and let C be a cycle on which all vertices of V are placed. For any $u, v \in V$, a *clockwise tour* (resp., *anti-clockwise tour*) is a part of the cycle C from u to v in a clockwise direction (resp., in an anti-clockwise direction). We use $C^{\mathrm{cl}}[u, v]$ (resp., $C^{\mathrm{acl}}[u, v]$) to denote this tour (Figure 1(a)). When we would like to specify the graph G, we use $C^{\mathrm{cl}}_G[u, v]$ (resp., $C^{\mathrm{acl}}_G[u, v]$). The tour $C^{\mathrm{cl}}[u, v]$, for example, can be expressed canonically as a sequence of vertices $(v_1, \ldots, v_k)$ such that $v_1 = u$, $v_k = v$, and $v_2, \ldots, v_{k-1}$ are all vertices visited along the cycle C clockwise. We use $C^{\mathrm{cl}}(u, v)$ and $C^{\mathrm{cl}}[u, v)$ (resp., $C^{\mathrm{acl}}(u, v)$ and $C^{\mathrm{acl}}[u, v)$) to denote the sub-sequences $(v_2, \ldots, v_{k-1})$ and $(v_1, \ldots, v_{k-1})$ respectively. Note here that it is not necessary that G has an edge between adjacent vertices in such a tour. The length of the tour is simply the number of vertices on the tour. An edge (u, v) of G is called a *chord* if u and v are not adjacent on the cycle C. For any chord (u, v), we may consider two arcs, namely, $C^{\mathrm{cl}}[u, v]$ and $C^{\mathrm{acl}}[u, v]$; but in the following, we will simply use $C[u, v]$ to denote one of them that is regarded as the arc of the chord (u, v) in the context. When necessary, we will state, e.g., "the arc $C^{\mathrm{cl}}[u, v]$" for specifying which one is currently regarded as the arc. A *gap-d* (resp., *gap-d^+*) *chord* is a chord (u, v) whose arc $C[u, v]$ is of length $d + 2$ (resp., length $\geq d + 2$). For any chord (u, v), the subplane inside of the cycle C surrounded by the chord (u, v) and the arc $C[u, v]$ is called the *lower area* of the chord; on the other hand, the other side of the chord within the cycle C is called the *upper area* (see Figure 1(a)). A *lowest gap-d^+* chord is a gap-d^+ chord that has no other gap-d^+ chord in its lower area. We say that two chords (u_1, v_1) and (u_2, v_2) *cross* if they cross in the circle C in a natural way (see Figure 1(b)). Formally, we say that (u_1, v_1) *crosses* (u_2, v_2) if either (i) u_2 is on the tour $C^{\mathrm{cl}}(u_1, v_1)$ and v_2 is on the tour $C^{\mathrm{acl}}(u_1, v_1)$, or (ii) v_2 is on the tour $C^{\mathrm{cl}}(u_1, v_1)$ and u_2 is on the tour $C^{\mathrm{acl}}(u_1, v_1)$. Also, we say that (u_1, v_1) *semi-crosses* (u_2, v_2) if either (i) u_2 is on the tour $C^{\mathrm{cl}}[u_1, v_1]$ and v_2 is on the tour $C^{\mathrm{acl}}[u_1, v_1]$, or (ii) v_2 is on the tour $C^{\mathrm{cl}}[u_1, v_1]$ and u_2 is on the tour $C^{\mathrm{acl}}[u_1, v_1]$ (see Figure 1(b)). Note that clearly crossing implies semi-crossing. In addition, we say that a chord (u_1, v_1) *separates* two chords (u_2, v_2) and (u_3, v_3) if the endpoints of two chords v_2 and

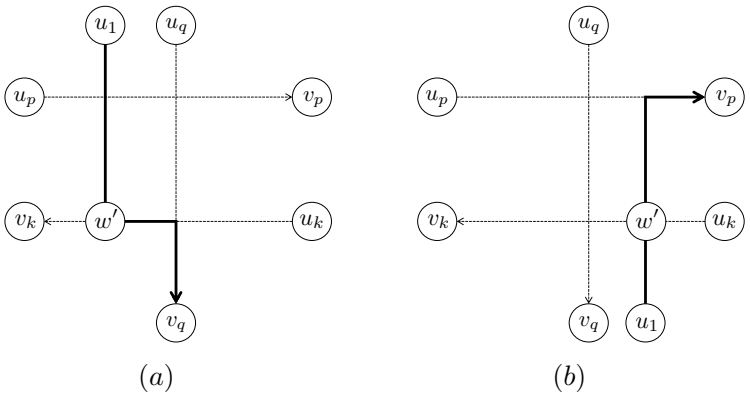


Figure 2 A common vertex w' of a path from u_k to v_k and a path from u_1 to v_q or v_p for some $p, q < k$.

v_3 are separated by the chord (u_1, v_1) (see Figure 1(c)). Formally, (u_1, v_1) *separates* (u_2, v_2) and (u_3, v_3) if either (i) v_2 is on the tour $C^{\mathrm{cl}}[u_1, v_1]$ and v_3 is on the tour $C^{\mathrm{acl}}[u_1, v_1]$, or (ii) v_3 is on the tour $C^{\mathrm{cl}}[u_1, v_1]$ and v_2 is on the tour $C^{\mathrm{acl}}[u_1, v_1]$. We say that k chords (u_1, v_1), $(u_2, v_2), \ldots, (u_k, v_k)$ are *traversable* if the following two conditions are satisfied:

1. (u_1, v_1) semi-crosses (u_2, v_2),
2. $\forall i \in [3, k]$, $\exists p, q < i$, (u_i, v_i) separates (u_p, v_p) and (u_q, v_q).

Now for the graph $G_0 = (V_0, E_0)$, we define the circle graph $G_0^{\mathrm{cir}} = (V_0^{\mathrm{cir}}, E_0^{\mathrm{cir}})$ by

$$\begin{aligned} V_0^{\mathrm{cir}} &= V_0^{\mathrm{rim}}, \text{ and} \\ E_0^{\mathrm{cir}} &= \left\{ (u, v) \,|\, \exists \text{path from } u \text{ to } v \text{ in } G_0 \right\}, \end{aligned}$$

where we assume that the rim vertices of V_0^{cir} $(= V_0^{\mathrm{rim}})$ are placed on a cycle C_0 as they are on the rim of the block in the grid graph. Then it is clear that G_0^{cir} keeps the same reachability relation among vertices in $V_0^{\mathrm{cir}} = V_0^{\mathrm{rim}}$. Recall that G_0 has $O(n^2)$ vertices. Thus, by using PlanarReach, we can show the following lemma.

▶ **Lemma 3.** *G_0^{cir} keeps the same reachability relation among vertices in $V_0^{\mathrm{cir}} = V_0^{\mathrm{rim}}$. That is, for any pair u, v of vertices of V_0^{cir}, v is reachable from u in G_0^{cir} if and only if it is reachable from u in G_0. There exists an algorithm that transforms G_0 to G_0^{cir} in $O(n)$-space and polynomial-time in n.*

The notion of traversable is a key for discussing the reachability on G_0^{cir}. Based on the following lemma, we use a traversable sequence of edges for characterizing the reachability on the circle graph G_0^{cir}.

▶ **Lemma 4.** *For a circle graph $G_0^{\mathrm{cir}} = (V_0^{\mathrm{cir}}, E_0^{\mathrm{cir}})$ obtained from a block grid graph G_0, if there are traversable edges (u_1, v_1), $(u_2, v_2), \ldots, (u_k, v_k) \in E_0^{\mathrm{cir}}$, then $(u_1, v_k) \in E_0^{\mathrm{cir}}$.*

Proof. We show that v_k is reachable from u_1 in G_0 by induction on k. First, we consider the case $k = 2$, namely (u_1, v_1) semi-crosses (u_2, v_2). G_0 contains a path p_{u_1,v_1} which goes from u_1 to v_1. Also, G_0 contains a path p_{u_2,v_2} which goes from u_2 to v_2. Since G_0 is planar and u_1, v_1, u_2, and v_2 are the rim vertices and the edges are semi-crossing, there exists a vertex w which is common in p_{u_1,v_1} and p_{u_2,v_2} in G_0. Since w is reachable from u and v_2 is reachable from w, there exists a path from u_1 to v_2.

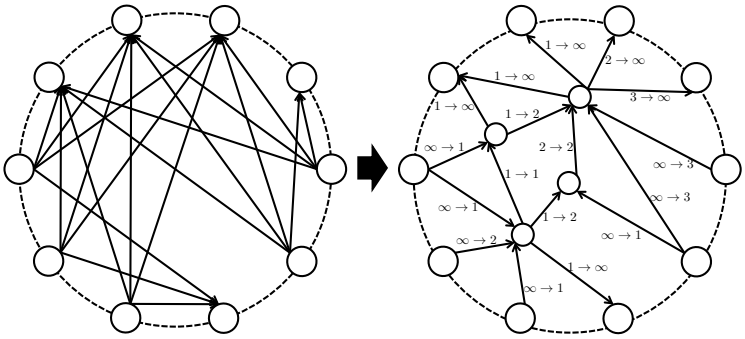


Figure 3 An example of the transformation from a circle graph to a gadget graph.

Next, we assume that the lemma is true for all sequences of traversable edges of length less than k. By the definition, there exist two edges (u_p, v_p) and (u_q, v_q) that the edge (u_k, v_k) separates $(p, q < k)$. We have two paths p_{u_1,v_p} from u_1 to v_p and p_{u_1,v_q} from u_1 to v_q in G_0 by the induction hypothesis. Also we have a path p_{u_k,v_k} from u_k to v_k. Since (u_k, v_k) separates (u_p, v_p) and (u_q, v_q), v_p and v_q are on the different sides of arcs of the edge (u_k, v_k). If u_1 and v_p are on the same arc of (u_k, v_k), the paths p_{u_1,v_q} and p_{u_k,v_k} have a common vertex w' (see Figure 2(a)). On the other hand, if u_1 and v_q are on the same arc of (u_k, v_k), the paths p_{u_1,v_p} and p_{u_k,v_k} have a common vertex w' (see Figure 2(b)). Thus there exists a path from u_1 to v_k via w' in G_0. ◀

3.2 Gadget graph

We introduce the notion of "gadget graph". A gadget graph is a graph that is given a "label set" to each edge.

▶ **Definition 5.** A *gadget graph* $\widetilde{G}$ is a graph defined by a tuple $(\widetilde{V}, \widetilde{E}, \widetilde{K}, \widetilde{L})$, where $\widetilde{V}$ is a set of vertices, $\widetilde{E}$ is a set of edges, $\widetilde{K}$ is a *path function* that assigns an edge or $\perp$ to each edge, and $\widetilde{L}$ is a *level function* that assigns a *label set* to each edge. A label set is a set $\{i_1 \to o_1, i_2 \to o_2, \ldots, i_k \to o_k\}$ of labels where each label $i_j \to o_j$, $i_j, o_j \in \mathbb{R} \cup \{\infty\}$, is a pair of *in-level* and *out-level*.

Remark. For an edge $(u, v) \in \widetilde{E}$, we may use expressions $\widetilde{K}(u, v)$ and $\widetilde{L}(u, v)$ instead of $\widetilde{K}((u, v))$ and $\widetilde{L}((u, v))$ for simplicity.

Our goal is to transform a given circle graph (obtained from a block grid graph) $G_0^{\mathrm{cir}} = (V_0^{\mathrm{cir}}, E_0^{\mathrm{cir}})$ in which all vertices in V_0^{cir} are placed on a cycle C to a *plane* gadget graph $\widetilde{G}_{\mathsf{p}} = (\widetilde{V}_{\mathsf{p}}^{\mathrm{out}} \cup \widetilde{V}_{\mathsf{p}}^{\mathrm{in}}, \widetilde{E}_{\mathsf{p}}, \widetilde{K}_{\mathsf{p}}, \widetilde{L}_{\mathsf{p}})$ where $\widetilde{V}_{\mathsf{p}}^{\mathrm{out}}$ is the set of *outer vertices* that are exactly the vertices of V_0^{cir} placed in the same way as G_0^{cir} on the cycle C, and $\widetilde{V}_{\mathsf{p}}^{\mathrm{in}}$ is the set of *inner vertices* placed inside of C. All edges of $\widetilde{E}_{\mathsf{p}}$ are also placed inside of C under our embedding. The inner vertices of $\widetilde{V}_{\mathsf{p}}^{\mathrm{in}}$ are used to replace crossing points of edges of E_0^{cir} to transform to a planar graph (see Figure 3). We would like to keep the "reachability" among vertices in $\widetilde{V}_{\mathsf{p}}^{\mathrm{out}}$ in $\widetilde{G}_{\mathsf{p}}$ while bounding $|\widetilde{V}_{\mathsf{p}}^{\mathrm{in}}| = O(n)$.

We explain how to characterize the reachability on a gadget graph. Consider any gadget graph $\widetilde{G} = (\widetilde{V}, \widetilde{E}, \widetilde{K}, \widetilde{L})$, and let x and y be any two vertices of $\widetilde{V}$. Intuitively, the reachability from x to y is characterized by a directed path on which we can send a token from x to y.

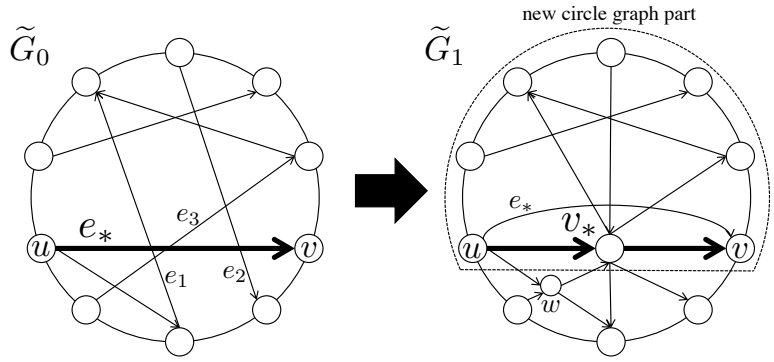


Figure 4 An initial transformation step from $\widetilde{G}_0$ to $\widetilde{G}_1$.

Suppose that there is a directed path $p = (e_1, \ldots, e_m)$ from x to y. We send a token through this path. The token has a level, which is initially ∞ when the token is at vertex x. (For a general discussion, we use a parameter ℓ_s for the initial level of the token.) When the token reaches the tail vertex $t(e_j)$ of some edge e_j of p with level ℓ, it can "go through" e_j to reach its head vertex $h(e_j)$ if $\widetilde{L}(e_j)$ has an *available label* $i_j \to o_j$ such that $i_j \leq \ell$ holds for its in-level i_j. If the token uses a label $i_j \to o_j$, then its level becomes the out-level o_j at the vertex $h(e_j)$. If there are several available labels, then we naturally use the one with the highest out-level. If the token can reach y in this way, we consider that a "token tour" from x to y is "realized" by this path p. Technically, we introduce $\widetilde{K}$ so that some edge can specify the next edge. We consider only a path $p = (e_1, \ldots, e_m)$ as "valid" such that $e_{i+1} = \widetilde{K}(e_i)$ for all e_i such that $\widetilde{K}(e_i) \neq \bot$. We characterize the reachability from x to y on gadget graph $\widetilde{G}$ by using a valid path realizing a token tour from x to y.

▶ **Definition 6.** For any gadget graph $\widetilde{G} = (\widetilde{V}, \widetilde{E}, \widetilde{K}, \widetilde{L})$, and for any two vertices x, y of $\widetilde{V}$, there exists a *token tour* from x to y with initial level ℓ_s if there exists a sequence of edges $(e_1, \ldots, e_m)$ that satisfies

1. $x = t(e_1)$ and $y = h(e_m)$,
2. $h(e_i) = t(e_{i+1})$ $(1 \leq i < m)$,
3. if $\widetilde{K}(e_i)$ is not $\bot$ $(1 \leq i < m)$, then $e_{i+1} = \widetilde{K}(e_i)$,
4. there exist labels $i_1 \to o_1 \in \widetilde{L}(e_1), \ldots, i_m \to o_m \in \widetilde{L}(e_m)$ such that $\ell_s \geq i_1$ and $o_t \geq i_{t+1}$ for all $1 \leq t < m$.

At the beginning of our algorithm, we obtain a gadget graph $\widetilde{G}_0 = (\widetilde{V}_0, \widetilde{E}_0, \widetilde{K}_0, \widetilde{L}_0)$ whose base graph is equal to G_0^{cir}, and $\widetilde{K}_0(e) = \bot$, $\widetilde{L}_0(e) = \{0 \to \infty\}$ for every $e \in \widetilde{E}_0$. It is obvious that G_0^{cir} and $\widetilde{G}_0$ have the same reachability. Namely, there exists a token tour from x to y for $x, y \in \widetilde{V}_0$ in $\widetilde{G}_0$ if and only if there exists an edge $(x, y) \in \widetilde{E}_0$.

We explain first the outline of our transformation from $\widetilde{G}_0$ to $\widetilde{G}_{\mathsf{p}}$. We begin by finding a chord $e_* = (u, v)$ with gap ≥ 2 having no other gap-2^+ chord in its lower area, that is, one of the lowest gap-2^+ chords. (If there is no gap-2^+ chord, then the transformation is terminated.) For this e_* and its lower area, we transform them into a planar part and reduce the number of crossing points as follows (see Figure 4): (i) Consider all edges of $\widetilde{G}_0$ crossing this chord e_* (e_1, e_2 and e_3 in Figure 4). Create a new inner vertex v_* of $\widetilde{G}_{\mathsf{p}}$ on the chord and bundle all crossing edges going through this vertex v_*; that is, we replace all edges crossing e_* by edges between their end points in the lower area of e_* and v_*, and edges between v_* and their end points in the upper area of e_*. (ii) Introduce new inner vertices for edges crossing

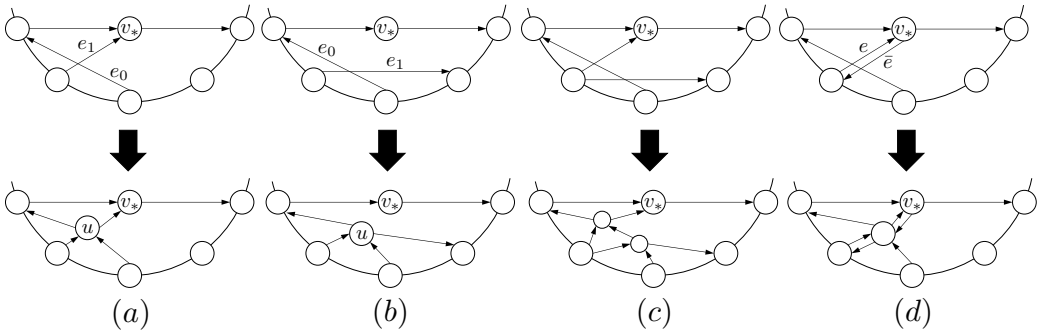


Figure 5 Examples of vertices made by MakePlanar.

gap-1 chords in the lower area of e_* (w in Figure 4). (iii) Add appropriate label sets to those newly introduced edges so that the reachability is not changed by this transformation. At this point we regard the lower area of e_* as processed, and remove this part from the circle graph part of $\widetilde{G}_0$ by replacing the arc $C[u, v]$ by a tour (u, v_*, v) to create a new circle graph part of $\widetilde{G}_1$. We then repeat this transformation step on the circle graph part of $\widetilde{G}_1$. In the algorithm, U_t is the vertices of the circle graph part of $\widetilde{G}_t$, thus $\widetilde{G}_t[U_t]$ indicates the circle graph part of $\widetilde{G}_t$. Note that e_* is not removed and becomes a gap-1 chord in the next step.

We explain step (ii) for $\widetilde{G}_0$ in more detail. Since e_* is a gap-2^+ chord, there exist only gap-1 chords or edges whose one end point is v_* in the lower area of e_*. If there are two edges e_0 and e_1 that cross each other, we replace the crossing point by a new inner vertex u (see Figure 5(a), (b)). The edge e_i becomes two edges $(t(e_i), u)$ and $(u, h(e_i))$ $(i = 0, 1)$, and we set $\widetilde{K}_1(t(e_i), u) = (u, h(e_i))$. The edges might be divided into more than two segments (see Figure 5(c)). We call the edge of $\widetilde{G}_0$ *original edge* of the divided edges. By the path function, we must move along the original edge. An edge e might have a reverse direction edge $\bar{e} = (h(e), t(e))$ (see Figure 5(d)). In this case, e and $\bar{e}$ share a new vertex for resolving crossing points. For $\widetilde{G}_t[U_t]$ $(t > 0)$, we process the lower area in the same way. We refer to this algorithm as MakePlanar, and the new inner vertices created by MakePlanar in step t as V^t_{MP}.

The detailed process of step (iii) is written in Algorithm 2, and Algorithm 1 describes the entire process of step (i), (ii) and (iii). The following lemma shows that an output graph of Algorithm 1 has small size.

▶ **Lemma 7.** *Algorithm 1 terminates creating a planar graph of size $O(n)$.*

Proof. In the beginning of the algorithm, $|U_0| = n$ and $|U_t|$ decreases by at least 1 for each iteration since the picked edge e^t_* is a gap-2^+ chord. Hence the algorithm stops after at most n iterations and the number of the new inner vertices made at line 7, or v^t_*, is also at most n. If a gap-k chord is picked, we make at most $2k - 1$ new inner vertices by MakePlanar, namely $|V^t_{\mathsf{MP}}| \leq 2k - 1$, since there exist only gap-1 chords in the lower area of the picked edge. The total number of inner vertices becomes at most

$$n + \sum_{i=1}^{t}(2k_i - 1) = n + 2\sum_{i=1}^{t} k_i - t \leq n + 2 \times 2n = 5n$$

where t is the number of iterations and k_i means that a gap-k_i chord was picked in the i-th iteration. After all, $|\widetilde{V}^{\mathrm{out}}_{\mathsf{p}} \cup \widetilde{V}^{\mathrm{in}}_{\mathsf{p}}| \leq n + 5n = 6n$. ◀

Algorithm 1

Input: A circle graph $G_0^{\mathrm{cir}} = (V_0^{\mathrm{cir}}, E_0^{\mathrm{cir}})$ obtained from a block graph.
Task: Output a plane gadget graph $\widetilde{G}_{\mathsf{p}} = (\widetilde{V}_{\mathsf{p}}^{\mathrm{out}} \cup \widetilde{V}_{\mathsf{p}}^{\mathrm{in}}, \widetilde{E}_{\mathsf{p}}, \widetilde{K}_{\mathsf{p}}, \widetilde{L}_{\mathsf{p}})$ which satisfies $\widetilde{V}_{\mathsf{p}}^{\mathrm{out}} = V_0^{\mathrm{cir}}$ and the reachability among vertices in $\widetilde{V}_{\mathsf{p}}^{\mathrm{out}}$ in $\widetilde{G}_{\mathsf{p}}$ is the same as G_0^{cir}.

1: initialize $t = 0$ // loop counter
2: $\widetilde{G}_0 = (\widetilde{V}^{\mathrm{out}} \cup \widetilde{V}_0, \widetilde{E}_0, \widetilde{K}_0, \widetilde{L}_0)$ where $\widetilde{V}^{\mathrm{out}} \leftarrow V_0^{\mathrm{cir}}, \widetilde{V}_0 \leftarrow \emptyset, \widetilde{E}_0 \leftarrow E_0^{\mathrm{cir}}, \widetilde{K}_0(e) \leftarrow \bot, \widetilde{L}_0(e) \leftarrow \{0 \to \infty\}$ for each $e \in E_0^{\mathrm{cir}}$, and $U_0 \leftarrow \widetilde{V}^{\mathrm{out}}$
3: for every $v \in \widetilde{V}^{\mathrm{out}}$, $\ell_{in}^0(v) \leftarrow 0, \ell_{out}^0(v) \leftarrow \infty, p^0(v) \leftarrow v$.
4: **while** $\widetilde{G}_t[U_t]$ has a lowest gap-2^+ chord **do**
5: pick a lowest gap-2^+ chord e_*^t
6: make a new vertex v_*^t
7: $\widetilde{V}_{t+1} \leftarrow \widetilde{V}_t \cup \{v_*^t\}$
8: $\widetilde{E}_{t+1} \leftarrow (\widetilde{E}_t \cup \{(t(e), v_*^t), (v_*^t, h(e)) \mid e \text{ crosses } e_* \text{ or } e = e_*\}) \setminus \{e \mid e \text{ crosses } e_*\}$
9: $U_{t+1} \leftarrow (U_t \cup \{v_*^t\}) \setminus C_{\widetilde{G}_t[U_t]}(t(e_*^t), h(e_*^t))$
10: use MakePlanar to make the lower area of e_*^t planar and update $\widetilde{V}_{t+1}$, $\widetilde{E}_{t+1}$ and $\widetilde{K}_{t+1}$.
11: change the labels by using Algorithm 2 for keeping reachability
12: output $\widetilde{G}_{t+1}[C_{\widetilde{G}_t[U_t]}[t(e_*^t), h(e_*^t)] \cup \{v_*^t\} \cup V_{\mathsf{MP}}^t]$, which is the lower area of e_*^t.
13: $t \leftarrow t + 1$
14: **end while**
15: use MakePlanar to make $\widetilde{G}_t[U_t]$ planar and assign labels by line 17-24 of Algorithm 2.
16: output $\widetilde{G}_t[U_t \cup V_{\mathsf{MP}}^t]$

Now we explain Algorithm 2 describing how to assign labels to $\widetilde{G}_{t+1}$ constructed in Algorithm 1. For each outer vertex $v \in \widetilde{V}^{\mathrm{out}}$, we keep three attributes $p^t(v)$, $\ell_{in}^t(v)$ and $\ell_{out}^t(v)$, and we call them *parent*, *in-level* and *out-level* respectively. We calculate these values from line 2 to 7 and line 25 to 27. $p^t(v)$ is a vertex belonging to the circle graph part of $\widetilde{G}_t$, namely $p^t(v) \in U_t$. From the algorithm, we can show that there are token tours from v to $p^t(v)$ and/or from $p^t(v)$ to v. For the token tour from v to $p^t(v)$, the final level of the token becomes $\ell_{in}^t(v)$. On the other hand, for the token tour from $p^t(v)$, it is enough to have $\ell_{out}^t(v)$ as an initial level to reach v. We will show these facts implicitly in the proof of Lemma 8.

At the beginning of each iteration of Algorithm 1, we choose a lowest gap-2^+ chord e_*^t. We collect vertices in U_t which are endpoints for some edges crossing with e_*^t, and we refer to the vertices among them which are in the lower area of e_*^t as S^ℓ and the vertices in the upper area of e_*^t as S^u (see Figure 6(a) and line 2). Next we collect vertices whose parents are in S^ℓ (resp., S^u), and we denote them by T^ℓ (resp., T^u) (line 3). Let x' and y' be vertices whose parents are $t(e_*^t)$ and $h(e_*^t)$ respectively. We assign indices to the vertices in T^u and T^ℓ such that the nearer to x' a vertex is located, the larger index the vertex has (see Figure 6(b)). We regard T^ℓ as a sequence $(t_1^\ell, t_2^\ell, \ldots, t_{|T^\ell|}^\ell)$, and T^u as a sequence $(t_1^u, t_2^u, \ldots, t_{|T^u|}^u)$. For each vertex t_i^ℓ in T^ℓ, we calculate $\ell_{in}^{t+1}(t_i^\ell)$ and $\ell_{out}^{t+1}(t_i^\ell)$ in Algorithm 3. From line 1 to 4, we decide temporary values of $\ell_{in}^{t+1}(t_i^\ell)$ and $\ell_{out}^{t+1}(t_i^\ell)$ according to reachability among vertices in T^ℓ and T^u in G_0^{cir}. When t_j^u has the maximum index among vertices that t_i^ℓ can reach in T^u, we let $\ell_{in}^{t+1}(t_i^\ell) = j + i/n$. When t_j^u has the minimum index among vertices which can reach t_i^ℓ in T^u, we let $\ell_{out}^{t+1}(t_i^\ell) = j + i/n$. The term i/n is for breaking ties. In the next for-loop, we change the in- and out-levels so that the in-level of the larger indexed vertex is larger than the out-level of the smaller indexed vertex. If there exists a vertex t_j^ℓ such that $i > j$ and $\ell_{out}^{t+1}(t_j^\ell) > \ell_{in}^{t+1}(t_i^\ell)$, then we let $\Delta = (\ell_{out}^{t+1}(t_j^\ell) - j/n) - (\ell_{in}^{t+1}(t_i^\ell) - i/n)$ and add Δ to $\ell_{in}^{t+1}(t_i^\ell)$ and $\ell_{out}^{t+1}(t_i^\ell)$. For preserving the magnitude relationship between in- and out-levels

Algorithm 2

Task: Set $\widetilde{L}_{t+1}$ so that $\widetilde{G}_{t+1}$ has the same reachability as $\widetilde{G}_t$

For every edge e appearing in both $\widetilde{G}_t$ and $\widetilde{G}_{t+1}$, let $\widetilde{L}_{t+1}(e) = \widetilde{L}_t(e)$.
S^ℓ (resp., S^u) $\leftarrow \{v \in U \mid \exists e \in \widetilde{E}_t$ s.t. e crosses e^t_*, $t(e) = v$ or $h(e) = v$, and v is at the lower (resp., upper) area of $e^t_*\}$
T^ℓ (resp., T^u) $\leftarrow \{v \in V_0^{\mathrm{cir}} \mid p^t(v) \in S^\ell$ (resp. $S^u)\}$
Fix any vertices $x', y' \in V_0^{\mathrm{cir}}$ such that $p^t(x') = t(e^t_*), p^t(y') = h(e^t_*)$.
Set an order to T^ℓ according to the order appearing in $C_{G_0^{\mathrm{cir}}}[y', x']$. We regard T^ℓ as a sequence $(t^\ell_1, t^\ell_2, \ldots, t^\ell_{|T^\ell|})$ (see Figure 6(b)).
Set an order to T^u in the same way as T^ℓ but according to the tour along the other arc. We also regard T^u as a sequence $(t^u_1, t^u_2, \ldots, t^u_{|T^u|})$ (see Figure 6(b)).
Use Algorithm 3 for calculating $\ell^{t+1}_{in}(v)$ and $\ell^{t+1}_{out}(v)$ for all $v \in T^\ell$.
for $u \in S^\ell$ **do**
  $\widetilde{L}_{t+1}(u, v^t_*) \leftarrow \{\ell^t_{in}(v) \to \ell^{t+1}_{in}(v) \mid p^t(v) = u\}$
  $\widetilde{L}_{t+1}(v^t_*, u) \leftarrow \{\ell^{t+1}_{out}(v) \to \ell^t_{out}(v) \mid p^t(v) = u\}$
end for
for $u \in S^u$ **do**
  $\widetilde{L}_{t+1}(u, v^t_*) \leftarrow \{\ell^t_{in}(t^u_i) \to \max_{t^\ell \in T^\ell}\{\ell^{t+1}_{out}(t^\ell) | \ (t^u_i, t^\ell) \in E_0^{\mathrm{cir}}\} | t^u_i \in T^u$ and $p^t(t^u_i) = u\}$
  $\widetilde{L}_{t+1}(v^t_*, u) \leftarrow \{\min_{t^\ell \in T^\ell}\{\ell^{t+1}_{in}(t^\ell) \mid (t^\ell, t^u_i) \in E_0^{\mathrm{cir}}\} \to \ell^t_{out}(t^u_i) | \ t^u_i \in T^u$ and $p^t(t^u_i) = u\}$
end for
$\widetilde{L}_{t+1}(t(e^t_*), v^t_*) \leftarrow \{\infty \to 0\}$, $\widetilde{L}_{t+1}(v^t_*, h(e^t_*)) \leftarrow \{\infty \to 0\}$
for all edge e created by MakePlanar **do**
  Let e' be the original edge of e
  if $t(e) = t(e')$ **then**
    $\widetilde{L}_{t+1}(e) \leftarrow \{a \to b \mid a \to b \in \widetilde{L}_t(e')\}$
  else
    $\widetilde{L}_{t+1}(e) \leftarrow \{b \to b \mid a \to b \in \widetilde{L}_t(e')\}$
  end if
end for
for $v \in \{w \in U_t \mid w$ is at the lower area of $e^t_*\}$ **do**
  $p^{t+1}(v) = v^t_*$
end for
Unchanged $\ell^t_{in}(\cdot), \ell^t_{out}(\cdot)$ and $p^t(\cdot)$ will be taken over to $\ell^{t+1}_{in}(\cdot), \ell^{t+1}_{out}(\cdot)$ and $p^{t+1}(\cdot)$.

Algorithm 3

Task: Calculate $\ell^{t+1}_{in}(v)$ and $\ell^{t+1}_{out}(v)$ for all $v \in T^\ell$.

for $i \in [1, |T^\ell|]$ **do**
  $\ell^{t+1}_{in}(t^\ell_i) \leftarrow \max\{j \mid (t^\ell_i, t^u_j) \in E_0^{\mathrm{cir}},\ t^u_j \in T^u\} + i/n$
  $\ell^{t+1}_{out}(t^\ell_i) \leftarrow \min\{j \mid (t^u_j, t^\ell_i) \in E_0^{\mathrm{cir}},\ t^u_j \in T^u\} + i/n$
end for
for $i = 1$ to $|T^\ell|$ **do**
  $\Delta \leftarrow \max(0, \max\{\ell^{t+1}_{out}(t^\ell_j) - j/n \mid 1 \le j < i\} - (\ell^{t+1}_{in}(t^\ell_i) - i/n))$
  for $k \in [i, |T^\ell|]$ **do**
    $\ell^{t+1}_{in}(t^\ell_k) \leftarrow \ell^{t+1}_{in}(t^\ell_k) + \Delta$
    $\ell^{t+1}_{out}(t^\ell_k) \leftarrow \ell^{t+1}_{out}(t^\ell_k) + \Delta$
  end for
end for

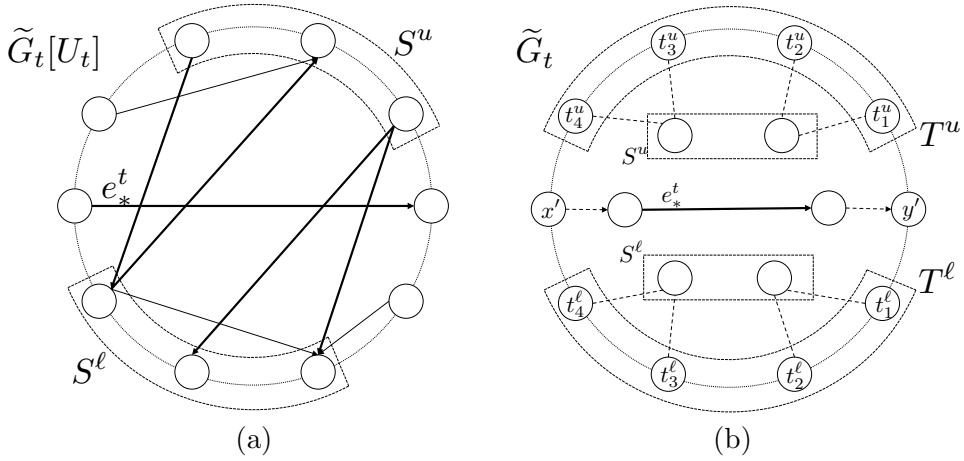


Figure 6 (a) An example of S^ℓ and S^u, (b) An example of T^ℓ and T^u.

of t^ℓ_i and those of t^ℓ_k $(k > i)$, we also add Δ to $\ell^{t+1}_{in}(t^\ell_k)$ and $\ell^{t+1}_{out}(t^\ell_k)$.

Back to Algorithm 2. From line 8 to 16, we assign labels to edges newly appearing in $\widetilde{G}_{t+1}$. Let v be any vertex in T^ℓ. For edges in the lower area of e^t_*, the edge $(p^t(v), v^t_*)$ has a label $\ell^t_{in}(v) \to \ell^{t+1}_{in}(v)$ (line 9), and the edge $(v^t_*, p^t(v))$ has a label $\ell^{t+1}_{out}(v) \to \ell^t_{out}(v)$ (line 10). Consider edges in the upper area of e^t_*. Let v be any vertex in T^u. The edge $(p^t(v), v^t_*)$ has a label $\ell^t_{in}(v) \to \ell_{max}$ where ℓ_{max} is the maximum in-level of vertices in T^ℓ that can reach v (line 13). The edge $(v^t_*, p^t(v))$ has a label $\ell_{min} \to \ell^t_{out}(v)$ where ℓ_{min} is the minimum out-level of vertices in T^ℓ that v can reach (line 14). The edges $(t(e^t_*), v^t_*)$ and $(v^t_*, h(e^t_*))$ have only one label $\infty \to 0$, which prohibits using these edges (line 16).

From line 17 to 24, we assign labels to edges made by MakePlanar. For every edge (u, v) in the lower area of e^t_*, the edge (u, v) might be divided into some edges, for instance $(u, w_1), (w_1, w_2), \ldots (w_k, v)$ by MakePlanar. In this case, when (u, v) has a label $a \to b$, (u, w_1) has a label $a \to b$ and the other edges have labels $b \to b$.

From line 25 to 27, we update the parents of the vertices in the lower area of e^t_*. For each vertex v in U_t and in the lower area of e^t_*, we let $p^{t+1}(v) = v^t_*$.

The following lemma shows that paths in $\widetilde{G}_0$ remain in $\widetilde{G}_t$ for every t.

▶ **Lemma 8.** *For any t in Algorithm 1, if there exists an edge from x toward y in $\widetilde{G}_0$, then there exists a token tour from x to y in $\widetilde{G}_t$ whose length is at most $2t + 1$.*

The following lemma shows the other direction: if there exists a token tour from x to y in the gadget graph, then there exists a path from x to y in the circle graph. From Lemma 4, it is enough to prove the following Lemma.

▶ **Lemma 9.** *For any t and $x, y \in V^{\mathrm{cir}}_0$, if there exists a token tour from x to y in $\widetilde{G}_t$, then there exists a traversable edge sequence $(e_1, \ldots, e_k)$ in G^{cir}_0 such that $t(e_1) = x$ and $h(e_k) = y$.*

We analyze the space and time complexity of Algorithm 1. Note that, for saving computation space, we do not implement the Algorithm straightforwardly in some points. We begin with the space complexity. We regard the circle graph $G^{\mathrm{cir}}_0 = (V^{\mathrm{cir}}_0, E^{\mathrm{cir}}_0)$ as the input. For every $v \in V^{\mathrm{cir}}_0$, we keep three attributes $\ell^t_{in}(v)$, $\ell^t_{out}(v)$ and $p^t(v)$ in step t. The in- and out-levels are rational numbers that have the form of $i + j/n$. Thus we keep two integers i and j for each in- and out-level. We use $\widetilde{O}(n)$ space for preserving them. In step

t, we also keep U_t by using $\widetilde{O}(n)$ space. We need $\widetilde{G}_t[U_t]$, but we do not keep $\widetilde{E}_t$ explicitly. For $u, v \in U_t$, whether there exists an edge (u, v) in $\widetilde{G}_t[U_t]$ is equivalent to whether there exists an edge (x, y) in E_0^{cir} such that $p^t(x) = u$ and $p^t(y) = v$. Since E_0^{cir} is included in the input, we could calculate it with $\widetilde{O}(1)$ space. We keep no other information throughout the Algorithm. The number of edges in $\widetilde{G}_t[U_t]$ is at most $2|U_t|^2 = O(n^2)$. Thus, for line 4 and 5, we can find a lowest gap-2^+ chord by using $\widetilde{O}(1)$ space. For line 7 and 9, we use only $\widetilde{O}(1)$ space for updating $\widetilde{V}_t$ and U_t. For line 8, we ignore the edges in the upper area of e_*^t (these edges belong to $\widetilde{G}_{t+1}[U_{t+1}]$, thus we have no need to keep them). For the edges in the lower area of e_*^t, since there exist only gap-1 chords in the area, the number of edges in the area is $O(n)$. We use $\widetilde{O}(n)$ space for temporarily keeping them. In MakePlanar (line 10), we look through them, and find crossing points and resolve them and set $\widetilde{K}_{t+1}(\cdot)$ by using $\widetilde{O}(n)$ space.

Now we consider Algorithm 2. The number of edges in $\widetilde{G}_t[U_t]$ is at most $2|U_t|^2 = O(n^2)$. Thus, for line 2, we can find S^ℓ and S^u by using $\widetilde{O}(1)$ space, and we use $\widetilde{O}(n)$ space for keeping them. For line 3 to 6, since $|T^\ell|, |T^u| = O(n)$, we also use $\widetilde{O}(n)$ for keeping T^ℓ and T^u. In addition, we use $\widetilde{O}(n)$ space for calculating $\ell_{in}^{t+1}(v)$ and $\ell_{out}^{t+1}(v)$ for all $v \in T^\ell$. In Algorithm 3, we use $\widetilde{O}(1)$ space for each operation and the length of for-loops is $O(n)$. Thus we use $\widetilde{O}(1)$ space in all. For line 8 to 11, we only refer to in- and out-levels that we are keeping. For line 12 to 15, we do not keep and ignore the labels belonging to edges in the upper area. For line 16, we use $\widetilde{O}(1)$ space. For line 17 to 24, we check whether an edge in the lower area was divided by MakePlanar and we use additional $\widetilde{O}(1)$ space. For line 25 to 27, we can find all vertices in the lower area of e_*^t by using $\widetilde{O}(n)$ space, and we use additional $\widetilde{O}(1)$ space for updating $p^{t+1}(\cdot)$.

We go back to Algorithm 1. For line 12, we output the information of the vertices, edges, labels and values of the path function in the lower area of e_*^t. Here we have to calculate the labels on the gap-1 chords (other information is preserved now). Let the gap-1 chord be (v_*^p, v_*^q). If $p < q$, this edge was made in step q and the labels on the edge were calculated at line 13 of Algorithm 2. Thus, for any $v \in V_0^{\mathrm{cir}}$ such that $p^t(v) = v_*^p$, we calculate $\ell_{out} = \max_{t^\ell \in V_0^{\mathrm{cir}}, p^t(t^\ell) = v_*^q} \{\ell_{out}^t(t^\ell) \mid (v, t^\ell) \in E_0^{\mathrm{cir}}\}$, and $\ell_{in}^t(v) \to \ell_{out}$ becomes one of the labels on the edge (if the vertex v is not in T^u, ℓ_{out} is not defined and a label for v does not exist). On the other hand, if $p > q$, this edge was made in step p and the labels on the edge were calculated at line 14 of Algorithm 2. Thus, for any $v \in V_0^{\mathrm{cir}}$ such that $p^t(v) = v_*^q$, we calculate $\ell_{in} = \min_{t^\ell \in V_0^{\mathrm{cir}}, p^t(t^\ell) = v_*^p} \{\ell_{in}^t(t^\ell) \mid (t^\ell, v) \in E_0^{\mathrm{cir}}\}$, and $\ell_{in} \to \ell_{out}^t(v)$ becomes one of the labels on the edge (if the vertex v is not in T^u, ℓ_{in} is not defined and a label for v does not exist). We use additional $\widetilde{O}(1)$ space for these calculation. For line 15, we trace line 10 to 12. In total, we use $\widetilde{O}(n)$ space.

Next consider the time complexity. In Lemma 7, we proved that the while-loop at line 4 stops after at most n steps. Since the sizes of U_t, S^ℓ, S^u, T^ℓ and T^u are all $O(n)$, every operation in the Algorithms takes poly(n) time. Thus this algorithm runs in polynomial time.

▶ **Lemma 10.** *Algorithm 1 runs in polynomial time with using $\widetilde{O}(n)$ space.*

From Lemma 8, 9 and 10, we can obtain $\widetilde{G}_{\mathsf{p}}$ with $\widetilde{O}(n) = \widetilde{O}(N^{1/3})$ space and polynomial time. By applying PlanarReach to the plane gadget graph with $O(N^{2/3})$ vertices, we can prove Theorem 2.

4 Conclusion

We presented an $\widetilde{O}(n^{1/3})$ space algorithm for the grid graph reachability problem. The most natural question is whether we can apply our algorithm to the planar graph reachability problem. Although the directed planar reachability is reduced to the directed reachability on grid graphs [1], the reduction blows up the size of the graph by a large polynomial factor and hence it is not useful. Moreover, it is known that there exist planar graphs that require quadratic grid area for embedding [10]. However we do not have to stick to grid graphs. We can apply our algorithm to graphs which can be divided into small blocks efficiently. For instance we can use our algorithm for king's graphs [6]. More directly, for using our algorithm, it is enough to design an algorithm that divides a planar graph into small blocks efficiently.

References

1 Eric Allender, David A Mix Barrington, Tanmoy Chakraborty, Samir Datta, and Sambuddha Roy. Planar and grid graph reachability problems. *Theory of Computing Systems*, 45(4):675–723, 2009.

2 Tetsuo Asano and Benjamin Doerr. Memory-constrained algorithms for shortest path problem. In *CCCG*, 2011.

3 Tetsuo Asano, David Kirkpatrick, Kotaro Nakagawa, and Osamu Watanabe. $\widetilde{O}(\sqrt{n})$-space and polynomial-time algorithm for planar directed graph reachability. In *International Symposium on Mathematical Foundations of Computer Science*, pages 45–56. Springer, 2014.

4 Greg Barnes, Jonathan F Buss, Walter L Ruzzo, and Baruch Schieber. A sublinear space, polynomial time algorithm for directed st connectivity. *SIAM Journal on Computing*, 27(5):1273–1282, 1998.

5 Chris Bourke, Raghunath Tewari, and NV Vinodchandran. Directed planar reachability is in unambiguous log-space. *ACM Transactions on Computation Theory (TOCT)*, 1(1):4, 2009.

6 Gerard Jennhwa Chang. Algorithmic aspects of domination in graphs. *Handbook of Combinatorial Optimization*, pages 221–282, 2013.

7 Tatsuya Imai, Kotaro Nakagawa, Aduri Pavan, NV Vinodchandran, and Osamu Watanabe. An $O(n^{1/2+\varepsilon})$-space and polynomial-time algorithm for directed planar reachability. In *Computational Complexity (CCC), 2013 IEEE Conference on*, pages 277–286. IEEE, 2013.

8 Omer Reingold. Undirected connectivity in log-space. *Journal of the ACM (JACM)*, 55(4):17, 2008.

9 Derrick Stolee and NV Vinodchandran. Space-efficient algorithms for reachability in surface-embedded graphs. In *Computational Complexity (CCC), 2012 IEEE 27th Annual Conference on*, pages 326–333. IEEE, 2012.

10 Leslie G Valiant. Universality considerations in vlsi circuits. *IEEE Transactions on Computers*, 100(2):135–140, 1981.

11 Avi Wigderson. The complexity of graph connectivity. *Mathematical Foundations of Computer Science 1992*, pages 112–132, 1992.

The Reverse Kakeya Problem

Sang Won Bae[1]
Kyonggi University
Suwon, Korea
swbae@kgu.ac.kr
https://orcid.org/0000-0002-8802-4247

Sergio Cabello[2]
University of Ljubljana and IMFM
Slovenia
sergio.cabello@fmf.uni-lj.si
https://orcid.org/0000-0002-3183-4126

Otfried Cheong[3]
KAIST
Daejeon, Korea
otfried@kaist.airpost.net
https://orcid.org/0000-0003-4467-7075

Yoonsung Choi
KAIST
Daejeon, Korea
giantsol2@kaist.ac.kr

Fabian Stehn
Bayreuth University
Germany
fabian.stehn@uni-bayreuth.de

Sang Duk Yoon[4]
Postech
Pohang, Korea
sangduk.yoon@gmail.com

Abstract

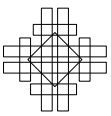
We prove a generalization of Pál's 1921 conjecture that if a convex shape P can be placed in any orientation inside a convex shape Q in the plane, then P can also be turned continuously through 360° inside Q. We also prove a lower bound of $\Omega(mn^2)$ on the number of combinatorially distinct maximal placements of a convex m-gon P in a convex n-gon Q. This matches the upper bound proven by Agarwal et al.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Kakeya problem, convex, isodynamic point, turning

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.6

[1] Supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2015R1D1A1A01057220).
[2] Supported by Slovenian Research Agency, program P1-0297.
[3] Supported by ICT R&D program of MSIP/IITP [R0126-15-1108].
[4] Supported by NRF grant 2011-0030044 (SRC-GAIA) funded by the government of Korea.



34th International Symposium on Computational Geometry (SoCG 2018).
Editors: Bettina Speckmann and Csaba D. Tóth; Article No. 6; pp. 6:1–6:13

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Acknowledgements This research was started during the Korean Workshop on Computational Geometry 2017, organized by Tohoku University in Zao Onsen. We thank all workshop participants for the helpful discussions. We also thank Xavier Goaoc and Helmut Alt for helpful discussions.

1 Introduction

In 1917, Soichi Kakeya posed the following problem: What is the minimum area region in the plane in which a needle of length 1 can be turned through 360° continuously, and return to its initial position [6]? For convex regions, the problem was solved by Pál [8], who showed that the solution is the equilateral triangle of height one, having area $1/\sqrt{3}$. For the general case, Besicovitch gave the surprising answer that one could rotate a needle using an arbitrary small area [2, 3]. Kakeya-type problems have received considerable attention in the literature, as there are strong connections to problems in number theory [4], geometric combinatorics [10], arithmetic combinatorics [7], oscillatory integrals, and the analysis of dispersive and wave equations [9].

If one generalizes the problem for convex regions slightly, and asks for the smallest convex region in which a given convex shape P can be turned through 360°, the problem seems to be still wide open: the answer is not even known when P is an equilateral triangle or a square.

In this paper, we consider a "reverse" version of the problem, where the convex compact shapes P and Q are already given, and we ask: how large can we make P such that it can turn through 360° inside Q?

Let's assume that the origin is in the interior of P, and denote P rotated by θ around the origin by P_θ. Being able to turn P inside Q obviously implies that P_θ can be translated into Q for any orientation θ. Is this condition also sufficient?

In his 1921 paper solving the convex case of the Kakeya problem, Pál [8] conjectured that this is the case. Intriguingly, the paper contains a footnote added during the proof stage, stating that Harald Bohr had proven this conjecture. Unfortunately, this proof seems to have never been published, and we have not been able to find another proof in the literature. We also do not know how exactly Pál defined "turning" in this context: Is the angle changing in a strictly monotone way, or merely monotonically?

We therefore prove a stronger version of Pál's conjecture: For a given angle θ, let $\lambda(\theta)$ be the largest scaling factor such that $\lambda(\theta)P_\theta$ can be translated into Q. We prove that the function $\theta \mapsto \lambda(\theta)$ is continuous, and show that there is a continuous function $\tau\colon [0, 2\pi] \to \mathbb{R}^2$ such that $\lambda(\theta)P_\theta + \tau(\theta) \subseteq Q$. In other words, P can be rotated, while continuously scaling and translating it to maintain the largest possible size that will fit inside Q at that orientation.

Our result implies Pál's conjecture: If P_θ can be translated into Q for any θ, then $\lambda(\theta)$ is always at least one, and so $P_\theta + \tau(\theta)$ is a continuous motion that turns P inside Q.

When P and Q are convex polygons, then our problem is closely related to work by Agarwal et al. [1]. They showed that the set of all *similar copies* of a convex m-gon P that lie in a given convex n-gon Q can be represented as a convex polytope $\mathcal{F}$ of complexity $O(mn^2)$ in $\mathbb{R}^4$. (A *similar copy* of a shape P is the image of P under scaling and translation.) One can project $\mathcal{F}$ onto a plane representing the scaling and rotation of P, and obtains a convex polygon $\mathcal{P}$, again of complexity $O(mn^2)$, whose boundary corresponds to points (θ, λ) where $\lambda P_\theta \subseteq Q$ and $\lambda = \lambda(\theta)$. In other words, the boundary of $\mathcal{P}$ immediately gives a description of the function $\theta \mapsto \lambda(\theta)$.

Agarwal et al. give a construction of a convex m-gon P and a convex n-gon Q such that there are $\Theta(mn^2)$ placements of similar copies of P inscribed into Q and realizing distinct

sets of vertex-edge contacts. This implies that the complexity of $\mathcal{F}$ is $\Theta(mn^2)$. However, their construction does not give a lower bound on the complexity of the projection $\mathcal{P}$ (which is equal to the complexity of the function $\theta \mapsto \lambda(\theta)$), since not all the placements in their construction are maximal.

We construct a convex m-gon P and a convex n-gon Q such that there are $\Theta(mn^2)$ maximal similar placements of P inscribed to Q. This implies that the complexity of $\mathcal{P}$, and therefore the complexity of the function $\theta \mapsto \lambda(\theta)$, is $\Theta(mn^2)$. For the special case of P being an equilateral triangle, our lower bound construction gives a convex n-gon Q that has $\Theta(n^2)$ combinatorially distinct inscribed maximal equilateral triangles. This implies that it may be difficult to improve on the quadratic-time algorithm for finding the largest equilateral triangle inscribed to a convex polygon by DePano et al. [5].

On the algorithmic side, Agarwal et al. give an $O(mn^2 \log n)$ time algorithm that computes $\mathcal{F}$ and its projection $\mathcal{P}$. From $\mathcal{F}$ and $\mathcal{P}$, we can construct the functions $\theta \mapsto \lambda(\theta)$ and $\theta \mapsto \tau(\theta)$. Given these functions, we can then answer questions such as:

1. *What is the largest similar copy of P inscribed into Q?* The answer to this question is given by the *maximum* of $\lambda(\theta)$, and its computation was the goal of Agarwal et al.
2. *What is the largest similar copy of P that can be turned through 360° inside Q?* This question is answered by the *minimum* of $\lambda(\theta)$.

To better understand the geometry of the problem, we give a purely geometric characterization of the local minima of $\theta \mapsto \lambda(\theta)$, leading to a necessary condition for the solution to question 2.

Finally, we consider the following problem: Given a triangle Q, how can we place one point on each of its edges such that the diameter of the resulting three-point set is minimized? We prove that if Q has no angle larger than 120°, then the answer is given by the corners of the largest equilateral triangle that can be turned inside Q (that is, the solution to question 2 above). This equilateral triangle can be found by constructing the first isodynamic point of Q.

2 Parameterization

Throughout this paper, P and Q are fixed convex compact shapes in the plane, with the origin contained in the interior of P. For $\theta \in [0, 2\pi]$, let P_θ be P rotated counter-clockwise by angle θ around the origin.

We will parameterize all the scaled, rotated, and translated placements of P as points in $\mathbb{R}^4$, as follows. For $\theta \in [0, 2\pi]$ and $\lambda \geqslant 0$, we define

$$s := \lambda \cos\theta,$$
$$t := \lambda \sin\theta.$$

For $(s, t, x, y) \in \mathbb{R}^4$, we define the affine map $\phi = \Phi(s, t, x, y) : \mathbb{R}^2 \mapsto \mathbb{R}^2$ as follows:

$$\phi(u) = \begin{pmatrix} s & t \\ -t & s \end{pmatrix} \cdot u + \begin{pmatrix} x \\ y \end{pmatrix} = \lambda \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \cdot u + \begin{pmatrix} x \\ y \end{pmatrix}.$$

In other words, ϕ rotates by θ around the origin, scales by λ, and finally translates by (x, y). In particular, $\phi(P) = \lambda P_\theta + (x, y)$.

We now define the set $\mathcal{F} \subset \mathbb{R}^4$ to correspond to all those ϕ such that $\phi(P) \subseteq Q$. Formally,

$$\mathcal{F} = \{(s, t, x, y) \mid \Phi(s, t, x, y)(P) \subseteq Q\}.$$

▶ **Lemma 1.** *The set $\mathcal{F}$ is a compact convex set in $\mathbb{R}^4$.*

Proof. For each point $p \in P$ and each closed halfspace h containing Q, define the set $H_{p,h} \subset \mathbb{R}^4$ of all the maps ϕ such that $\phi(p) \in h$. Since Φ is an affine map and h is a halfspace, $\phi(p) \in h$ is a linear constraint in the variables s, t, x, y, and so $H_{p,h}$ is a closed halfspace in $\mathbb{R}^4$.

We claim that $\mathcal{F} = \bigcap H_{p,h}$. Obviously, for any $\phi \in \mathcal{F}$, $p \in P$, and $h \supset Q$ we have $\phi(p) \in h$, so it remains to prove the other direction. Consider a $\phi \notin \mathcal{F}$. This means that $\phi(P)$ contains a point $\phi(p) \notin Q$. Since Q is a compact convex set, this implies that there is a halfspace h containing Q but not containing $\phi(p)$. It follows that $\phi \notin H_{p,h}$.

As the intersection of closed convex sets, $\mathcal{F}$ is closed and convex. Since $s^2 + t^2 = \lambda^2$ and for sufficiently large λ we can never translate λP into Q (for instance, when λ is larger than the ratio of the diameters of the two shapes), s and t are bounded. Finally, for $(x, y) \notin Q$, we always have $\phi(P) \not\subseteq Q$ because P contains the origin, and so x and y are bounded. ◀

We now define the set $\mathcal{P} \subset \mathbb{R}^2$ as the projection of $\mathcal{F}$ onto the (s, t)-plane. By Lemma 1, $\mathcal{P}$ is a compact convex set in the plane. Note that a pair (s, t) corresponds uniquely to a pair (λ, θ) and lies in $\mathcal{P}$ if and only if there are $(x, y) \in \mathbb{R}^2$ such that $(s, t, x, y) \in \mathcal{F}$. In other words, whenever λP_θ can be translated into Q.

The origin lies in the interior of $\mathcal{P}$, since for small enough λ we can always translate P_θ into Q (for instance if λ is smaller than the ratio of the largest inscribed circle of Q and the smallest circumscribed circle of P).

Consider now a fixed $\theta \in [0, 2\pi]$. The set of (s, t) corresponding to (λ, θ), for $\lambda \geqslant 0$, is the ray from the origin with orientation θ in the (s, t)-plane. Since $\mathcal{P}$ is convex and the origin lies in its interior, this ray intersects the boundary of $\mathcal{P}$ in a single point (s_θ, t_θ). We set $\lambda(\theta) := \sqrt{s_\theta^2 + t_\theta^2}$. By definition of $\mathcal{F}$ and $\mathcal{P}$, $\lambda(\theta)$ is the largest scaling factor such that $\lambda(\theta)P_\theta$ can be translated into Q.

▶ **Lemma 2.** *The function $\lambda\colon [0, 2\pi] \to \mathbb{R}$ is continuous.*

Proof. We show that $\theta \mapsto (s_\theta, t_\theta)$ is continuous, this implies the lemma. Assume for a contradiction, that there is a θ and an $\varepsilon > 0$ such that for every $i > 0$ there exists a θ_i with $|\theta - \theta_i| < 1/i$ such that the distance between (s_θ, t_θ) and $(s_{\theta_i}, t_{\theta_i})$ is at least ε. The points $(s_{\theta_i}, t_{\theta_i})$ all lie on the boundary of $\mathcal{P}$, so compactness of $\mathcal{P}$ implies that there is a subsequence that converges to some (s, t) on the boundary of $\mathcal{P}$. This means we can write $(s, t) = (s_{\theta'}, t_{\theta'})$ for some $\theta' \neq \theta$. Since the origin lies in the interior of $\mathcal{P}$, there is a disk of radius $\rho > 0$ around the origin that lies inside $\mathcal{P}$. Let $\delta = |\theta - \theta'|$. For $i > 2/\delta$, we have $|\theta_i - \theta'| > \delta/2$. Since $(s_{\theta_i}, t_{\theta_i})$ and $(s_{\theta'}, t_{\theta'})$ both have distance at least ρ from the origin, their distance is lower-bounded in terms of ρ and δ, contradicting the existence of the subsequence converging to $(s_{\theta'}, t_{\theta'})$. ◀

To simplify the notation, let's define $P_\theta^* = \lambda(\theta)P_\theta$. Our goal is to find a continuous function $\tau\colon [0, 2\pi] \to \mathbb{R}^2$ such that $P_\theta^* + \tau(\theta) \subseteq Q$.

3 The polygonal case

In this section we will assume that P is a convex m-gon, while Q is a convex n-gon. The set $\mathcal{F}$ can then be described as the intersection of only $n \times m$ halfspaces in $\mathbb{R}^4$. Indeed, to ensure that $\phi(P) \subseteq Q$ it suffices to require that for each *vertex* p of P we have $\phi(p) \in Q$, which is equivalent to require that $\phi(p) \in h$ for each halfplane h supporting an edge of Q.

By the upper-bound theorem, $\mathcal{F}$ is thus a convex polytope in $\mathbb{R}^4$ of complexity $O(n^2m^2)$. In fact, Agarwal et al. [1] have shown that $\mathcal{F}$ has complexity $O(mn^2)$, and that it can be computed in time $O(mn^2 \log n)$.

The projection $\mathcal{P}$ is therefore a convex polygon with at most $O(mn^2)$ vertices. As we have seen in the previous section, its boundary encodes the function $\lambda(\theta)$.

It remains to define the continuous function $\theta \mapsto \tau(\theta)$. We proceed as follows: Each vertex v of $\mathcal{P}$ is of the form (s_θ, t_θ) for some θ. This gives us the value of $\lambda(\theta)$ for this θ. The vertex v is the projection of at least one vertex $(s_\theta, t_\theta, x_\theta, y_\theta)$ of $\mathcal{F}$. We pick one such vertex and define $\tau(\theta) = (x_\theta, y_\theta)$. Finally, we interpolate linearly between these values along the segments connecting consecutive vertices in $\mathbb{R}^4$. By convexity of $\mathcal{F}$, these segments lie in $\mathcal{F}$ and are therefore feasible. Since each segment projects on an edge of $\mathcal{P}$, each point on a segment corresponds indeed to a translation of P^*_θ. The resulting function $\tau\colon [0, 2\pi] \to \mathbb{R}^2$ is continuous.

We observe that Agarwal et al. gave a simpler algorithm to compute the projection $\mathcal{P}$ in time $O(mn^2 \log n)$ without computing $\mathcal{F}$ first. It seems that this simpler algorithm is not sufficient for our purposes, as we need a vertex of $\mathcal{F}$ corresponding to each vertex of $\mathcal{P}$.

▶ **Theorem 3.** *Given a convex m-gon P and a convex n-gon Q, we can in time $O(mn^2 \log n)$ compute the continuous function $\lambda(\theta)$ and a continuous function $\tau\colon [0, 2\pi] \to \mathbb{R}^2$ such that $P^*_\theta + \tau(\theta) = \lambda(\theta)P_\theta + \tau(\theta) \subseteq Q$.*

4 The general case

The goal of this section is to generalize Theorem 3 to compact convex shapes. We start by investigating the nature of the set of feasible translations for a given angle θ.

▶ **Lemma 4.** *For each $\theta \in [0, 2\pi]$, the set of feasible translations $T(\theta) := \{\tau \in \mathbb{R}^2 \mid P^*_\theta + \tau \subseteq Q\}$ is either a point or a line segment.*

Proof. Since P^*_θ and Q are convex, the set $T(\theta) = \{\tau \in \mathbb{R}^2 \mid P^*_\theta + \tau \subseteq Q\}$ is convex. If $T(\theta)$ is neither a point nor a line segment, then we can find three distinct points $p, q, r \in T(\theta)$ that are not collinear. Set $\sigma = (p + q + r)/3$, and observe that $P^*_\theta + \sigma$ lies strictly in the interior of Q. It follows that there is a $\delta > 0$ such that $(1 + \delta)P^*_\theta + \sigma \subseteq Q$, contradicting the definition of $\lambda(\theta)$. ◀

▶ **Lemma 5.** *For any $\theta \in [0, 2\pi]$ where $T(\theta)$ is a line segment, the boundary of Q contains two line segments e, e' that are parallel to the line segment $T(\theta)$. For any point σ in the relative interior of $T(\theta)$ there is a $\delta > 0$ such that for any θ' with $|\theta - \theta'| < \delta$ there exists an $\sigma' \in T(\theta')$ such that the line $\sigma\sigma'$ is orthogonal to $T(\theta)$.*

Proof. We assume that $T(\theta)$ is horizontal, and its endpoints are $(0, 0)$ and $\tau_1 = (x_1, 0)$. Among the points with the largest y-coordinate in P^*_θ, let p_1 be the leftmost one, and let p_2 be the rightmost one (where $p_1 = p_2$ is possible as well). Among the points with the smallest y-coordinate in P^*_θ, let q_1 and q_2 be the leftmost and the rightmost one. Since $P^*_\theta \subset Q$ and $P^*_\theta + \tau_1 \subset Q$, the convex hull $\mathcal{C}$ of P^*_θ and $P^*_\theta + \tau_1$ also lies in Q.

Let now σ be any point in the relative interior of $T(\theta)$, that is $\sigma = (x, 0)$ with $0 < x < x_1$. The set $P^*_\theta + \sigma$ lies in $\mathcal{C}$, and touches the boundary of $\mathcal{C}$ only on the horizontal segments $p_1(p_2 + \tau_1)$ and $q_1(q_2 + \tau_1)$. Thus, by definition of $\lambda(\theta)$, there must be a boundary point of Q on the relative interior of both these segments. But $\mathcal{C} \subseteq Q$ implies that this is only possible if both $p_1(p_2 + \tau_1)$ and $q_1(q_2 + \tau_1)$ are part of the boundary of Q, forming the segments e and e'.

Let $\mathcal{H}$ be the strip bounded by the horizontal lines through e and e', let $\mathcal{D}$ be the closure of $\mathcal{H} \setminus \mathcal{C}$, and let $P_0 := P_\theta^* + \sigma$. Since $\mathcal{D} \cap P_0 = \emptyset$, $\mathcal{D}$ is closed, and P_0 is compact, there is a $\rho > 0$ such that any point in $\mathcal{D}$ and any point in P_0 have distance at least ρ.

Let D be the diameter of P_0, and let W be the width of the strip $\mathcal{H}$. Let

$$\delta < \min\Big(\frac{W}{4D}, \frac{\rho}{8D}, \frac{\rho W}{32D^2}\Big),$$

and consider θ' with $|\theta - \theta'| < \delta$. We will show that there is a y such that the translation $\sigma' = (x, y) \in T(\theta')$.

We first obtain $P_1 := \lambda(\theta)P_{\theta'} + \sigma$ from $P_0 = P_\theta^* + \sigma$ by a rotation around point σ by an angle smaller than δ. Let W' be the vertical width of the set P_1. We scale P_1 by factor W/W' around σ, obtaining $P_2 = \frac{W}{W'}\lambda(\theta)P_{\theta'} + \sigma$ of vertical width W. Finally, we translate P_2 vertically by vector $(0, y)$ such that $P_3 = P_2 + (0, y) \subset \mathcal{H}$.

We claim that $P_3 = \frac{W}{W'}\lambda(\theta)P_{\theta'} + (x, y) \subset Q$. Since the vertical width of P_3 is W, this implies $\lambda(\theta') = \frac{W}{W'}\lambda(\theta)$, and so $P_3 = P_{\theta'}^* + (x, y)$, and the claim $\sigma' = (x, y) \in T(\theta')$ will follow.

During the rotation, each point of P_θ^* travels a distance of at most δD. This implies that the Hausdorff distance of P_0 and P_1 is at most $\delta D < \rho/8$. Therefore

$$W - 2\delta D \leqslant W' \leqslant W + 2\delta D, \qquad \text{and so}$$
$$1 + \frac{2\delta D}{W - 2\delta D} \geqslant \frac{W}{W'} \geqslant 1 - \frac{2\delta D}{W + 2\delta D}$$

Using $\delta \leqslant W/(4D)$ we have $2\delta D \leqslant W/2$, and so

$$1 + 4\delta\frac{D}{W} \geqslant \frac{W}{W'} \geqslant 1 - \frac{4}{3}\delta\frac{D}{W}, \qquad \text{implying}$$
$$4\delta\frac{D}{W} \geqslant \frac{W}{W'} - 1 \geqslant -\frac{4}{3}\delta\frac{D}{W}, \qquad \text{and so} \qquad \Big|\frac{W}{W'} - 1\Big| \leqslant 4\delta\frac{D}{W}.$$

During the scaling by factor W/W', a point travels a distance of at most $|W/W' - 1| \cdot D$, which is bounded by $4\delta D^2/W < \rho/8$.

It follows that P_2 and P_0 have Hausdorff distance at most $\rho/8 + \rho/8 = \rho/4$. This implies in particular that the translation length $\tau \leqslant \rho/4$, which in turn implies that the Hausdorff distance of P_3 and P_0 is at most $\rho/2$.

There is no point of $\mathcal{D}$ within distance $\rho/2$ of P_0, so $P_3 \cap \mathcal{D} = \emptyset$. From $P_3 \subset \mathcal{H}$ then follows $P_3 \subset \mathcal{C} \subset Q$. ◀

Let $\mathcal{S}(\theta)$ be the two-dimensional affine subspace of $\mathbb{R}^4$ where the first two coordinates are $\lambda(\theta)\cos\theta$ and $\lambda(\theta)\sin\theta$. In other words,

$$\mathcal{S}(\theta) := \big\{(\lambda(\theta)\cos\theta, \lambda(\theta)\sin\theta, x, y) \,\big|\, (x, y) \in \mathbb{R}^2\big\}.$$

Since the first two coordinates are constant, $\mathcal{S}(\theta)$ is parallel to the (x, y)-plane in $\mathbb{R}^4$. We next set $\mathcal{T}(\theta) := \mathcal{S}(\theta) \cap \mathcal{F}$. Note that $T(\theta)$ is the projection of $\mathcal{T}(\theta)$ on the (x, y)-plane, or, put differently, $\mathcal{T}(\theta)$ is $T(\theta)$ "lifted" to the two-dimensional affine subspace $\mathcal{S}(\theta)$.

We define the function $\ell\colon [0, 2\pi] \to \mathbb{R}$ such that $\ell(\theta)$ is the length of the segment $T(\theta)$ (and zero when $T(\theta)$ is a point). Since $\mathcal{S}(\theta)$ is parallel to the (x, y)-plane, $\ell(\theta)$ is also the length of the segment $\mathcal{T}(\theta)$. Finally, we define the function $m\colon [0, 2\pi] \to \mathbb{R}^4$ such that $m(\theta)$ is the midpoint of the segment $\mathcal{T}(\theta)$.

▶ **Lemma 6.** *The function $[0, 2\pi] \mapsto m(\theta)$ is continuous.*

Proof. Let's assume for a contradiction that the claim is false. Then there exists a $\theta \in [0, 2\pi]$ and an $\varepsilon > 0$ such that for every $i \in \{1, 2, \dots\}$ there is $\theta_i \in [0, 2\pi]$ such that $|\theta - \theta_i| < 1/i$ and $|m(\theta) - m(\theta_i)| > \varepsilon$.

Let p_i and q_i be the endpoints of $\mathcal{T}(\theta_i)$. These points lie in the set $\mathcal{S} \cap \mathcal{F}$, where $\mathcal{S} = \bigcup_{\theta \in [0,2\pi]} \mathcal{S}(\theta)$. Since $\mathcal{S}$ is closed and $\mathcal{F}$ is compact, the intersection $\mathcal{S} \cap \mathcal{F}$ is compact, and so (p_i) and (q_i) have converging subsequences. To avoid double indices, we replace our sequence (θ_i) by such a subsequence where both (p_i) and (q_i) converge, and set $p := \lim_{i\to\infty} p_i$ and $q := \lim_{i\to\infty} q_i$. By compactness, $p, q \in \mathcal{S} \cap \mathcal{F}$. Since $p_i, q_i \in \mathcal{S}(\theta_i)$ and $\lim_{i\to\infty} \theta_i = \theta$, we have $p, q \in \mathcal{S}(\theta)$, and so $p, q \in \mathcal{S}(\theta) \cap \mathcal{F} = \mathcal{T}(\theta)$. By continuity of vector addition, the midpoint $m(\theta_i)$ of $p_i q_i$ converges to the midpoint m of pq, that is, $m = \lim_{i\to\infty} m(\theta_i)$. Because $|m(\theta) - m(\theta_i)| > \varepsilon$, we have $|m(\theta) - m| \geq \varepsilon$. Since $m(\theta)$ and m are on $\mathcal{T}(\theta)$, it follows that the segment $\mathcal{T}(\theta)$ has length at least ε.

We now pick points $(s_\theta, t_\theta, x_1, y_1)$ and $(s_\theta, t_\theta, x_2, y_2)$ from the segment $\mathcal{T}(\theta)$ at distance $\varepsilon/3$ from the two endpoints, respectively, where $s_\theta = \lambda(\theta)\cos\theta$ and $t_\theta = \lambda(\theta)\sin\theta$. Let $\sigma_1 := (x_1, y_1)$ and $\sigma_2 := (x_2, y_2)$ be the projections of the two points onto $T(\theta)$. By Lemma 5, there exists a $\delta > 0$ such that for any $\theta' \in [0, 2\pi]$ with $|\theta - \theta'| < \delta$ there exist points $\sigma_1', \sigma_2' \in T(\theta')$ such that $\sigma_1'\sigma_1$ and $\sigma_2'\sigma_2$ are orthogonal to $T(\theta)$. Since the segment $\sigma_1'\sigma_2'$ is a subset of $T(\theta')$, this implies that

$$\ell(\theta') \geqslant |\sigma_1'\sigma_2'| \geqslant |\sigma_1\sigma_2| = \ell(\theta) - \frac{2\varepsilon}{3}.$$

It follows that for $i > 1/\delta$, we have

$$|p_i q_i| = \ell(\theta_i) \geqslant \ell(\theta) - \frac{2\varepsilon}{3}.$$

which implies that the segment pq has length at least $\ell(\theta) - 2\varepsilon/3$.

Since the length of $\mathcal{T}(\theta)$ is $\ell(\theta)$ and pq is a subset of $\mathcal{T}(\theta)$, the distance between the midpoint m of pq and the midpoint $m(\theta)$ of $\mathcal{T}(\theta)$ is at most $\varepsilon/3$. This is a contradiction to the assumption that $|m(\theta) - m(\theta_i)| > \varepsilon$ for all i. ◀

We now obtain our theorem by defining $\tau(\theta)$ to be the midpoint of $T(\theta)$, that is, the projection of $m(\theta)$ on the (x, y)-plane.

▶ **Theorem 7.** *For any compact convex shapes P and Q, there exists a continuous function $\tau\colon [0, 2\pi] \to \mathbb{R}^2$ such that $P_\theta^* + \tau(\theta) = \lambda(\theta)P_\theta + \tau(\theta) \subseteq Q$.*

Using the same proof idea as in Lemma 6, we can also show that the function $\theta \mapsto \ell(\theta)$ is continuous.

5 Lower bounds

We now construct convex polygons P and Q such that there are many combinatorially distinct inscribed maximal similar placements of P in Q. These imply identical lower bounds for the complexity of the polygon $\mathcal{P}$.

▶ **Theorem 8.** *For any n there is a convex n-gon Q such that there are $\Theta(n^2)$ maximal placements of an equilateral triangle in Q.*
For any m and n there is a convex m-gon P and a convex n-gon Q such that there are $\Theta(mn^2)$ maximal similar placements of P in Q.

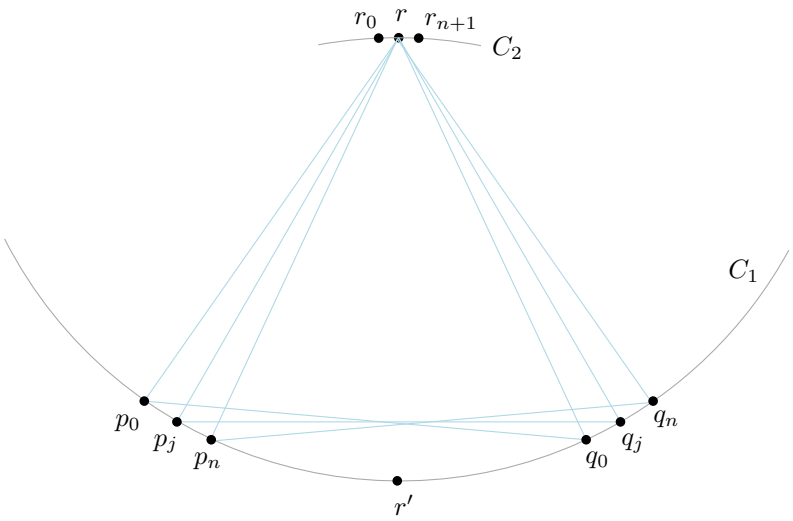


Figure 1 Construction of Q with many equilateral triangles inscribed.

Proof. Let rr' be a vertical segment of length one. Let C_1 be a unit radius circle with center r, let C_2 be a unit radius circle with center r', see Figure 1. Let $\beta = \pi/36$ and assume $n \equiv 2 \pmod 4$. We distribute $n+1$ points $p_0, p_1, \ldots, p_n$ regularly along the arc of C_1 from polar coordinate $4\pi/3 - \beta$ to polar coordinate $4\pi/3 + \beta$, and $n+1$ points $q_0, q_1, \ldots, q_n$ regularly along the arc of C_1 from polar coordinate $5\pi/3 - \beta$ to polar coordinate $5\pi/3 + \beta$ (Figure 1 shows only one intermediate point). Finally, we place $n+2$ points $r_0, \ldots, r_{n+1}$ on an arc of length ε of C_2 around r (The points are exaggerated in the figure, as they would otherwise be indistinguishable from r.)

We now construct a polygon Q as the intersection of three groups of halfplanes as follows:

- the lower halfplanes with bounding lines $r_0r_1, r_1r_2, r_2r_3, \ldots, r_{n-1}r_n$;
- the upper halfplanes with bounding lines $p_0p_2, p_5p_6, p_9p_{10}, p_{13}p_{14}, \ldots, p_{n-1}p_n$;
- the upper halfplanes with bounding lines $q_0q_1, q_4q_5, q_8q_9, q_{12}q_{13}, \ldots, q_{n-2}q_n$.

We observe that r, p_j, q_j form an equilateral triangle of side length one for each $j \in \{0, 1, \ldots, n\}$. The points p_1 and q_{n-1} lie outside Q, the points $p_3, p_4, p_7, p_8, p_{11}, p_{12}, \ldots p_{n-2}$ and the points $q_2, q_3, q_6, q_7, q_{10}, q_{11}, \ldots q_{n-3}$ lie in the interior of Q, and all other points lie on the boundary of Q.

When ε is small enough, then the circles of radius one around each r_i intersect the boundary of Q in the same edges and in points very close to the intersection points with the circle C_1. This implies that for every $i \in \{1, 2, \ldots, n\}$ and every $j \in \{3, 7, 11, 15, \ldots, n-3\}$, there is an equilateral triangle $\triangle_{ij}$ with side length one with one corner at r_i and two other corners very close to p_j and q_j in the interior of Q. If we rotate $\triangle_{ij}$ clockwise around r_i, it will ultimately hit the edge supported by the line $p_{j-2}p_{j-1}$ in a point close to p_{j-1}, see Figure 2. If we rotate $\triangle_{ij}$ counter-clockwise around r_i, then it will ultimately hit the edge supported by the line $q_{j+1}q_{j+2}$ in a point close to q_{j+1}. Let $f(\theta)$ and $g(\theta)$ be the length of the ray from r_i to the boundary of Q along the two incident edges of $\triangle_{ij}$, as we rotate along the interval where the triangle is contained in Q. Each function evaluates to one at one extreme and to a value strictly larger than one at the other extreme, implying that there must be an intermediate angle where $f(\theta) = g(\theta) > 1$. At this orientation we can enlarge $\triangle_{ij}$ to an equilateral triangle of side length $f(\theta) = g(\theta)$ that touches the boundary of Q at every corner. One corner is at r_i, the second touches an edge supported either by $p_{j-2}p_{j-1}$

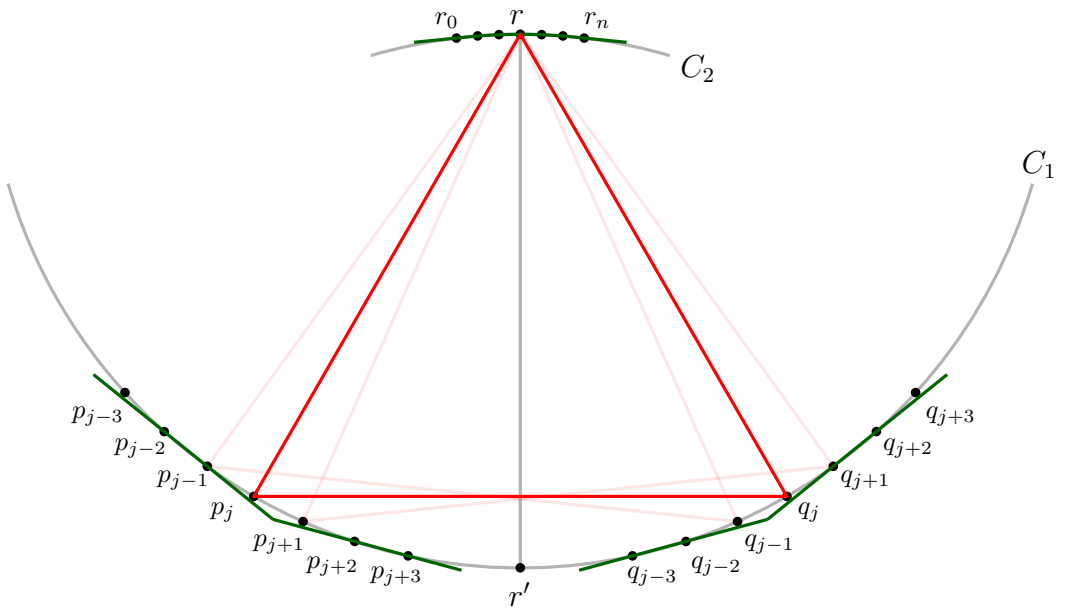


Figure 2 Detail of the construction of Q (magnified).

or $p_{j+2}p_{j+3}$, and the third touches an edge supported either by $q_{j-3}q_{j-2}$ or by $q_{j+1}q_{j+2}$. It follows that there are $\Theta(n^2)$ combinatorially distinct such equilateral triangles. Each of them is maximal for the given orientation, since the slopes of the incident edges do not allow for the triangle to be translated in any way.

We will now generalize the construction to a convex m-gon P. We first observe that the construction above will work for any sufficiently small value of β. The polygon P is obtained by taking an equilateral triangle $\triangle wu_0v_0$ of side length one, rotating it m times around w by an angle $\alpha < \pi/(6n)$, and taking the convex hull of these $m+1$ equilateral triangles $\triangle wu_kv_k$, see the red polygon in Figure 3. The polygon Q is obtained as above, but with $\beta < \alpha/12$. For every $k \in \{0, 1, \ldots, m\}$, every $i \in \{1, \ldots, n\}$, and every $j \in \{3, 7, 11, 15, \ldots, n-3\}$, there is a placement of P such that w coincides with r_i, u_k is very close to p_j in the interior of Q, v_k is very close to q_j in the interior of Q, and all other vertices of P lie well inside the interior of Q. As in the previous construction, we can now slightly rotate and scale this placement to obtain a maximal placement of P. ◀

6 A geometric construction for triangles and the isodynamic point

We consider now the following question: Given convex polygons P and Q, what is the largest scaling factor λ^* such that $\lambda^* P$ can be fully turned inside Q? From the considerations of Section 2 follows that $\lambda^* = \min_{0 \leqslant \theta < 2\pi} \lambda(\theta)$. In particular, λ^* is a local minimum of $\theta \mapsto \lambda(\theta)$. We now give a geometric characterization of these local minima that will also allow us to construct them.

First we consider the case when P and Q are triangles. For a fixed θ, $\lambda(\theta)$ can be computed by linear programming, with three variables λ, x, y for the translation and scaling, and 3×3 constraints to keep all vertices of $\lambda P + (x, y)$ inside Q. This means that there is an optimal solution where at least three constraints are tight: either the three vertices of $\lambda P + (x, y)$ lie on the three edges of Q, or a vertex of $\lambda P + (x, y)$ coincides with a vertex of Q and another vertex of $\lambda P + (x, y)$ lies on an edge of Q.

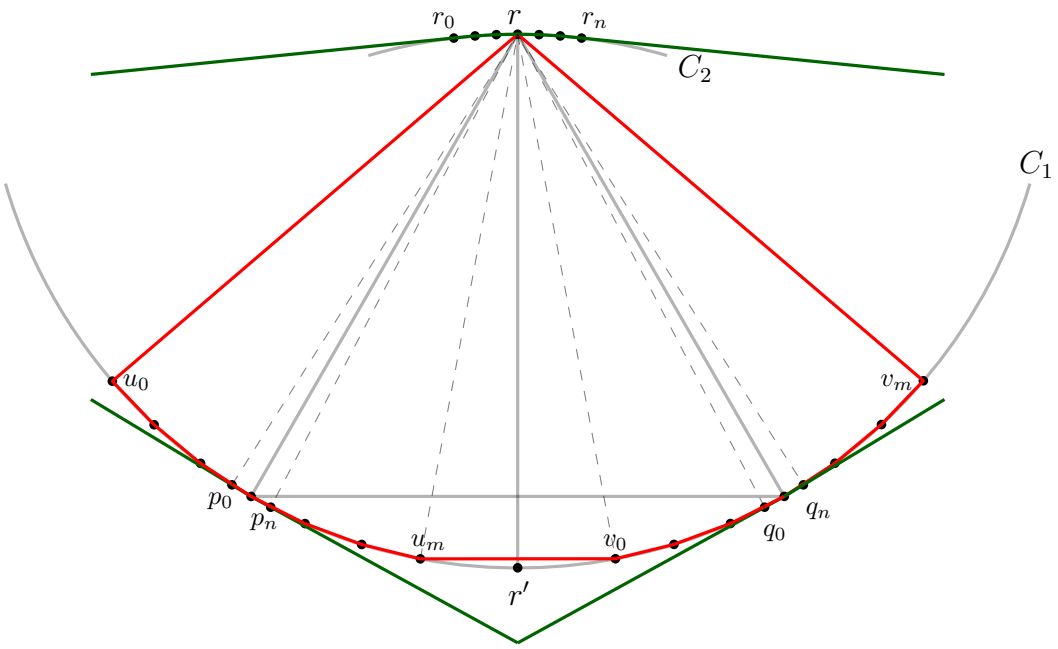


Figure 3 Generalizing the lower bound to a convex m-gon P.

Consider now an angle $\theta_0 \in [0, 2\pi]$ where λ has a local minimum, and let $\lambda_0 = \lambda(\theta_0)$. That implies there is an $\varepsilon > 0$ such that for $\theta_0 - \varepsilon < \theta < \theta_0 + \varepsilon$ we have $\lambda(\theta) \geqslant \lambda_0$. In other words, for $\theta \in [\theta_0 - \varepsilon, \theta_0 + \varepsilon]$, we can translate $\lambda_0 P_\theta$ into Q.

If a vertex U of $\lambda_0 P_{\theta_0} + (x, y)$ coincides with a vertex of Q and another vertex V lies on an edge e of Q, this is only possible if the segment uv is orthogonal to e. In other words, an edge of $\lambda_0 P_{\theta_0} + (x, y)$ coincides with a height of Q. There are nine possible candidate placements of this form.

Otherwise, the three vertices U, V, W of $\lambda_0 P_{\theta_0} + (x, y)$ must lie on the three edges a, b, c of Q. We have the following lemma.

▶ **Lemma 9.** *If the vertices U, V, W of a triangle $\triangle$ lie on the three edges a, b, c of a triangle Q, but there exists an $\varepsilon > 0$ such that $\triangle_\theta$ can be translated into Q for all $-\varepsilon < \theta < \varepsilon$, then the normals to a, b, c in U, V, W meet in a point inside the circumcircle of Q. There is at most one such placement that is similar to a given triangle P.*

Proof. Assume the normals do not meet in a point. Then pick a point F in the interior of the only bounded cell in the arrangement of normals. Rotating $\triangle$ around F will cause all vertices of $\triangle$ to leave Q, either for the clockwise or counter-clockwise rotation. So, for arbitrarily small θ, there is a triangle $\triangle_\theta$ that cannot be translated into Q, a contradiction.

Let now A, B, C be the vertices of Q such that $a = BC$, $b = AC$ and $c = AB$. Let α, β, and γ be the triangle angles incident to A, B and C, respectively. We are looking for a point $F \in \mathbb{R}^2$ such that the orthogonal projections of F onto the edges a, b, and c form a triangle $\bar{A}\bar{B}\bar{C}$ that is similar to the given triangle $P = \triangle UVW$, or, in other words, so that $\bar{A}\bar{B} : \bar{B}\bar{C} = UV : VW$ and $\bar{A}\bar{B} : \bar{A}\bar{C} = UV : UW$. We first observe that if F does not lie in the circumcircle of Q, then its orthogonal projection on at least one of the supporting lines of the edges of Q does not lie on the edge, so F must indeed lie inside the circumcircle.

Since $\angle A\bar{B}F = \angle A\bar{C}F = 90°$, the points $\bar{B}$ and $\bar{C}$ lie on the circle with diameter AF, and so we have $\bar{B}\bar{C} = AF \sin\alpha$. With the same reasoning we obtain $\bar{C}\bar{A} = BF \sin\beta$ and

$\bar{A}\bar{B} = CF \sin\gamma$. This means that the point F is the point where the following holds:

$$UV : VW = \bar{A}\bar{B} : \bar{B}\bar{C} = CF \sin\gamma : AF \sin\alpha$$
$$UV : UW = \bar{A}\bar{B} : \bar{A}\bar{C} = CF \sin\gamma : BF \sin\beta$$

It follows that the point F satisfies

$$AF : CF = VW \sin\gamma : UV \sin\alpha$$
$$BF : CF = UW \sin\gamma : UV \sin\beta$$

The points X with constant $AX : CX$ lie on an Apollonius circle around a center H on the line AC. The center lies outside the segment AC (and the circle degenerates to the bisector of A and C when $AX : CX = 1 : 1$). Similarly, the points X with constant $BX : CX$ lie on a circle around a center K on the line BC, outside the segment BC. The two circles intersect in at most two points F_1 and F_2. The two points are inverses of each other with respect to the circumcircle of Q, so only one of them can lie inside the circumcircle. ◀

Lemma 9 allows us to construct three more candidate placements for λP_θ inside Q (for the three circular assignments of vertices of P to the edges of Q). In total, we have 12 candidate placements, and the smallest feasible placement determines the bottleneck scaling factor λ^*.

An interesting special case of Lemma 9 occurs when P is an equilateral triangle. In this case the point F of the lemma is a well-known triangle center, the *first isodynamic point*. The two isodynamic points of a triangle $\triangle ABC$ (only the equilateral triangle has only one) can be defined as the points F whose orthogonal projections on the lines supporting the triangle edges form an equilateral triangle, or equivalently as the points F such that $AF : BF : CF = 1/a : 1/b : 1/c$.

The geometric construction of Lemma 9 can also be applied to arbitrary convex polygons P and Q. Again, the linear programming argument shows that for each θ, the placement of $P^*_\theta = \lambda(\theta)P_\theta$ in Q satisfies three constraints with equality, so either three corners of P^*_θ lie on three edges of Q, or a vertex of P^*_θ coincides with a vertex of Q and another vertex of P^*_θ lies on an edge of Q. We can thus look at all possible triples of corners of P and triples of edges of Q, and geometrically construct candidate placements. The smallest feasible placement again determines λ^*.

7 Minimizing the diameter of points on the edges of a triangle

Consider now the following problem: Given a triangle $Q = \triangle ABC$, find three points u, v, w, one on each edge of Q, such that the diameter of the set $\{u, v, w\}$ is minimized. Since the diameter of three points, one on each edge of Q, is a continuous function defined on a compact domain, there exists a set of three points u, v, w that realizes the minimum diameter. Let us denote this minimal diameter as $\nabla(Q)$.

We first claim that $\nabla(Q) \leqslant \lambda^* = \min_{0 \leqslant \theta < 2\pi} \lambda(\theta)$, where λ is our well-known function for P an equilateral triangle of side length one and the given Q.

Indeed, we saw in Section 6 that λ^* is attained either by an equilateral triangle that touches each edge of Q, or by a height of Q. In the first case we immediately get a candidate solution $\{u, v, w\}$ of cost λ^*. In the second case, let's assume it's the height h_A defined by A and its projection H_A on the edge $a = BC$. Then we can choose $u = H_A$ and pick v and w close to A on b and c for a candidate solution of cost smaller than λ^*.

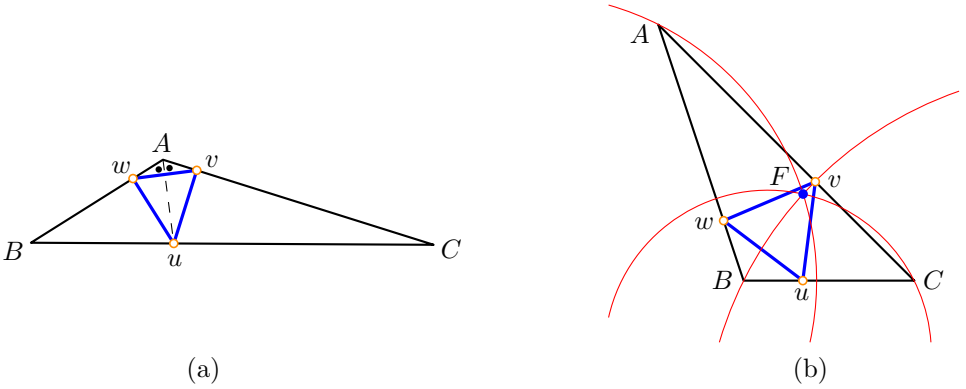


Figure 4 Point sets of minimal diameter.

▶ **Theorem 10.** *If triangle $Q = \triangle ABC$ has no angle larger than 120°, then $\nabla(Q) = \lambda^*$. If its angle $\alpha > 120°$, then $\nabla(Q)$ is given by the points $\{u, v, w\}$, where u is the intersection of the angular bisector of α with the edge $a = BC$, and v and w are the orthogonal projections of u onto $b = AC$ and $c = AB$.*

Proof. Let's first assume that $\alpha > 120°$, and let u, v, w be as stated (see Figure 4 (a)). Since $\alpha > 120°$, we have $\angle vuw = 180° - \alpha < 60°$, and so $vw < uv = uw$, implying that the diameter of $\{u, v, w\}$ is $uv = uw$. Assume for a contradiction there is a point set $u' \in a$, $v' \in b$, $w' \in c$ whose diameter is at most uv. If u' lies between u and C, then its distance to the line AB is larger than $uw = uv$; if u' lies between B and u, then its distance to the line AC is larger than uv. In both cases we obtain a contradiction, so $u' = u$. Since v and w are the only points on b and c at distance at most uv from u, we also have $v' = v$ and $w' = w$, proving the second part of the theorem.

Assume now that Q has no angle larger than 120°, and let u, v, w be a point set attaining the optimal diameter $\nabla(Q)$. Since (by an argument as above) $\nabla(Q)$ is less than the shortest height of Q, none of u, v, w coincides with a vertex of Q.

We claim that u, v, w form an isosceles triangle such that the angle between the two equal sides is at most 60°. Assume for a contradiction that uv is the unique longest edge. Then the diameter of the set $\{u, v, w\}$ is the length of uv, however, the length of uv can be reduced by moving u and v slightly. It follows that there is not a unique longest edge, and thus u, v, w must form an isosceles triangle such that the angle between the two equal sides is at most 60°.

Without loss of generality, we can assume that $uv = uw \geqslant vw$, and that $u \in a = BC$, $v \in b = AC$, $w \in c = AB$. We next show that the triangle with vertices u, v, w is equilateral. Assume for a contradiction that $vw < uv = uw$, which implies $\angle vuw < 60°$. If uv is not perpendicular to b, then we can reduce the length of uv by moving v slightly, and then we can move u slightly to be closer to w. The same argument holds for uw, so $uv \perp b$ and $uw \perp c$, which implies $\alpha = 180° - \angle vuw > 120°$, leading to a contradiction.

It follows that $vw = uv = uw$, so the points u, v, w form an equilateral triangle. Since $\triangle uvw$ is inscribed to Q, its side length is $\lambda(\theta)$ for some θ, so $\nabla(Q) \leqslant \lambda^*$ implies $\nabla(Q) = \lambda^*$, see Figure 4 (b). ◀

For $\alpha < 120°$, λ^* is a local minimum of the form of Lemma 9, so we can find the solution by projecting the first isodynamic point on each triangle edge. If $\alpha = 120°$, then the intersection of the angular bisector of α with the edge a is the first isodynamic point, and both constructions coincide.

References

1. P. K. Agarwal, N. Amenta, and M. Sharir. Largest placement of one convex polygon inside another. *Discrete & Computational Geometry*, 19:95–104, 1998. doi:10.1007/PL00009337.
2. A. S. Besicovitch. Sur deux questions de l'intégrabilité. *Journal de la Société des Math. et de Phys.*, II, 1920.
3. A. S. Besicovitch. On Kakeya's problem and a similar one. *Math. Zeitschrift*, 27:312–320, 1928.
4. J. Bourgain. Harmonic analysis and combinatorics: How much may they contribute to each other? In V. I. Arnold, M. Atiyah, P. Lax, and B. Mazur, editors, *Mathematics: Frontiers and Perspectives*, pages 13–32. American Math. Society, 2000.
5. A. DePano, Yan Ke, and J. O'Rourke. Finding largest inscribed equilateral triangles and squares. In *Proc. 25th Allerton Conf. Commun. Control Comput.*, 1987.
6. S. Kakeya. Some problems on maxima and minima regarding ovals. *The Science Report of the Tohoku Imperial University, Series 1, Mathematics, Physics, Chemistry*, 6:71–88, 1917.
7. I. Laba. From harmonic analysis to arithmetic combinatorics. *Bulletin (New Series) of the AMS*, 45:77–115, 2008.
8. G. Pál. Ein Minimumproblem für Ovale. *Math. Ann.*, 83:311–319, 1921.
9. T. Tao. From rotating needles to stability of waves: Emerging connections between combinatorics, analysis and PDE. *Notices of the AMS*, 48:297–303, 2001.
10. T. Wolff. Recent work connected with the Kakeya problem. In H. Rossi, editor, *Prospects in Mathematics*. American Math. Society, 1999.

Capacitated Covering Problems in Geometric Spaces

Sayan Bandyapadhyay
Department of Computer Science, University of Iowa
Iowa City, USA
sayan-bandyapadhyay@uiowa.edu

Santanu Bhowmick
Department of Computer Science, University of Iowa
Iowa City, USA
santanu-bhowmick@uiowa.edu

Tanmay Inamdar
Department of Computer Science, University of Iowa
Iowa City, USA
tanmay-inamdar@uiowa.edu

Kasturi Varadarajan
Department of Computer Science, University of Iowa
Iowa City, USA
kasturi-varadarajan@uiowa.edu

Abstract

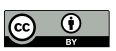
In this article, we consider the following capacitated covering problem. We are given a set P of n points and a set $\mathcal{B}$ of balls from some metric space, and a positive integer U that represents the *capacity* of each of the balls in $\mathcal{B}$. We would like to compute a subset $\mathcal{B}' \subseteq \mathcal{B}$ of balls and assign each point in P to some ball in $\mathcal{B}'$ that contains it, such that the number of points assigned to any ball is at most U. The objective function that we would like to minimize is the cardinality of $\mathcal{B}'$.

We consider this problem in arbitrary metric spaces as well as Euclidean spaces of constant dimension. In the metric setting, even the uncapacitated version of the problem is hard to approximate to within a logarithmic factor. In the Euclidean setting, the best known approximation guarantee in dimensions 3 and higher is logarithmic in the number of points. Thus we focus on obtaining "bi-criteria" approximations. In particular, we are allowed to expand the balls in our solution by some factor, but optimal solutions do not have that flexibility. Our main result is that allowing constant factor expansion of the input balls suffices to obtain constant approximations for this problem. In fact, in the Euclidean setting, only $(1 + \epsilon)$ factor expansion is sufficient for any $\epsilon > 0$, with the approximation factor being a polynomial in $1/\epsilon$. We obtain these results using a unified scheme for rounding the natural LP relaxation; this scheme may be useful for other capacitated covering problems. We also complement these bi-criteria approximations by obtaining hardness of approximation results that shed light on our understanding of these problems.

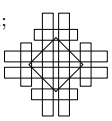
2012 ACM Subject Classification Theory of computation → Packing and covering problems, Theory of computation → Rounding techniques, Theory of computation → Computational geometry, Mathematics of computing → Approximation algorithms

Keywords and phrases Capacitated covering, Geometric set cover, LP rounding, Bi-criteria approximation

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.7



34th International Symposium on Computational Geometry (SoCG 2018).
Editors: Bettina Speckmann and Csaba D. Tóth; Article No. 7; pp. 7:1–7:15



Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Related Version A full version of the paper is available at https://arxiv.org/abs/1707.05170

Funding This material is based upon work supported by the National Science Foundation under Grants CCF-1318996 and CCF-1615845

1 Introduction

In this paper, we consider the following capacitated covering problem. We are given a set P of n points and a set $\mathcal{B}$ of balls from some metric space, and a positive integer U that represents the *capacity* of each of the balls in $\mathcal{B}$. We would like to compute a subset $\mathcal{B}' \subseteq \mathcal{B}$ of balls and assign each point in P to some ball in $\mathcal{B}'$ that contains it, such that the number of points assigned to any ball is at most U. The objective function that we would like to minimize is the cardinality of $\mathcal{B}'$. We call this the *Metric Capacitated Covering* (MCC) problem.

An important special case of this problem arises when $U = \infty$, and we refer to this as *Metric Uncapacitated Covering* (MUC). The MUC requires us to cover the points in P using a minimum number of balls from $\mathcal{B}$, and we can therefore solve it using the efficient greedy algorithm for set cover and obtain an approximation guarantee of $O(\log n)$. The approximation factor of $O(\log n)$ for Set Cover cannot be improved unless $P = NP$ [14]. The same is true for the MUC, as demonstrated by the following reduction from Set Cover. We construct a graph with a vertex corresponding to each set and element, and an edge of length 1 between a set and an element if the element is contained in the set. In the metric induced by this graph, we create an MUC instance: we include a ball of radius 1 at each set vertex, and let the points that need to be covered be the element vertices. It is easy to see that any solution for this instance of MUC directly gives a solution for the input instance of the general Set Cover, implying that for MUC, it is not possible to get any approximation guarantee better than the $O(\log n)$ bound for Set Cover.

The MUC in fixed dimensional Euclidean spaces has been extensively studied. One interesting variant is when the allowed set $\mathcal{B}$ of balls consists of all *unit* balls. Hochbaum and Maass [19] gave a polynomial time approximation scheme (PTAS) for this variant using a grid shifting strategy. When $\mathcal{B}$ is an arbitrary finite set of balls, the problem seems to be much harder. An $O(1)$ approximation algorithm in the 2-dimensional Euclidean plane was given by Brönnimann and Goodrich [9]. More recently, a PTAS was obtained by Mustafa and Ray [26]. In dimensions 3 and higher, the best known approximation guarantee is still $O(\log n)$. Motivated by this, Har-Peled and Lee [18] gave a PTAS for a bi-criteria version where the algorithm is allowed to expand the input balls by a $(1 + \epsilon)$ factor. Covering with geometric objects other than balls has also been extensively studied; see [22, 12, 4, 27, 10, 17] for a sample.

The MCC is a special case of the *Capacitated Set Cover* (CSC) problem. In the latter problem, we are given a set system $(X, \mathcal{F})$ with $n = |X|$ elements and $m = |\mathcal{F}|$ subsets of X. For each set $\mathcal{F}_i \in \mathcal{F}$, we are also given an integer U_i, which is referred to as its *capacity*. We are required to find a minimum size subset $\mathcal{F}' \subseteq \mathcal{F}$ and assign each element in X to a set in $\mathcal{F}'$ containing it, such that for each set $\mathcal{F}_i$, the number of points assigned to $\mathcal{F}_i$ is at most U_i. The MCC is obtained as a special case of CSC by setting $X = P$, $\mathcal{F} = \mathcal{B}$, and $U_i = U$ $\forall i$. Set Cover is a special case of CSC where the capacity of each set is ∞.

Applications of Set Cover include placement of wireless sensors or antennas to serve clients, VLSI design, and image processing [7, 19]. It is natural to consider capacity constraints that appear in many applications, for instance, an upper bound on the number of clients that

can be served by an antenna. Such constraints lead to the natural formulation of CSC. For the CSC problem, Wolsey [28] used a greedy algorithm to give an $O(\log n)$ approximation. For the special case of vertex cover (where each element in X belongs to exactly two sets in $\mathcal{F}$), Chuzhoy and Naor [11] presented an algorithm with approximation ratio 3, which was subsequently improved to 2 by Gandhi et al [15]. The generalization where each element belongs to at most a bounded number f of sets has been studied in a sequence of works, culminating in [20, 29]. Berman et al [7] have considered the "soft" capacitated version of the CSC problem that allows making multiple copies of input sets. Another closely related problem to the CSC problem is the so-called *Replica Placement* problem. For the graphs with treewidth bounded by t, an $O(t)$ approximation algorithm for this problem is presented in [1]. Finally, PTASs for the Capacitated Dominating Set, and Capacitated Vertex Cover problems on the planar graphs are presented in [6], under the assumption that the demands and capacities of the vertices are upper bounded by a constant.

Compared to MUC, relatively fewer geometric versions of the MCC problem have been studied in the literature. We refer to the version of MCC where the underlying metric is Euclidean as the *Euclidean Capacitated Covering* (ECC) problem. The dimension of the Euclidean space is assumed to be a constant. One such version arises when $\mathcal{B}$ comprises of all possible unit balls. This problem appeared in the Sloan Digital Sky Survey project [25]. Building on the shifting strategy of Hochbaum and Maass [19], Ghasemi and Razzazi [16] obtain a PTAS for this problem. When the set $\mathcal{B}$ of balls is arbitrary, the best known approximation guarantee is $O(\log n)$, even in the plane.

Given this state of affairs for MCC and ECC, we focus our efforts on finding a bi-criteria approximation. Specifically, we allow the balls in our solution to expand by at most a constant factor λ, without changing their capacity constraints (but optimal solution does not expand). We formalize this as follows. An (α, β)-approximation for a version of MCC, is a solution in which the balls may be expanded by a factor of β (i.e. for any ball B_i, and any point $p_j \in P$ that is assigned to B_i, $d(c_i, p_j) \leq \beta \cdot r_i$), and its cost is at most α times that of an optimal solution (which does not expand the balls). From the reduction of Set Cover to MUC described above, we can see that it is NP-hard to get an $(f(n), \lambda)$-approximation for any $\lambda < 3$ and $f(n) = o(\log n)$. This follows from the observation that in the constructed instance of MUC, the distance between a set vertex and a vertex corresponding to an element not in that set is at least 3. We note that it is a common practice in the wireless network setting to expand the radii of antennas at the planning stage to improve the quality of service. For example, Bose et al [8] propose a scheme for replacing omni-directional antennas by directional antennas that expands the antennas by a constant factor.

Related work. Capacitated version of facility location and clustering type problems have been well-studied over the years. One such clustering problem is the capacitated k-center. In the version of this problem with uniform capacities, we are given a set P of points in a metric space, along with an integer capacity U. A feasible solution to this problem is a choice of k centers to open, together with an assignment of each point in P to an open center, such that no center is assigned more than U points. The objective is to minimize the maximum distance of a point to its assigned center. $O(1)$ approximations are known for this problem [5, 21]; the version with non-uniform capacities is addressed in [2, 13]. Notice that the decision version of the uniform capacitated k-center with the radius parameter r is the same as the decision version of a special case of MCC, where the set $\mathcal{B}$ consists of balls of radius r centered at each point of the capacitated k-center instance. The capacity of each ball is the same as the uniform capacity U of the points. We want to find whether there is a

subset of $\mathcal{B}$ consisting of k balls that can serve all the points without violating the capacity constraint. For capacitated versions of other related optimization problems such as metric facility location, k-median etc, see [3, 23, 24] for recent advances.

1.1 Our results and contributions.

In this article, we make significant progress on both the MCC and ECC problems.

- We present an $(O(1), 6.47)$-approximation for the MCC problem. Thus, if we are allowed to expand the input balls by a constant factor, we can obtain a solution that uses at most $O(1)$ times the number of balls used by the optimal solution. As noted above, if we are not allowed to expand by a factor of at least 3, we are faced with a hardness of approximation of $\Omega(\log n)$.
- We present an $(O(\epsilon^{-4d} \log(1/\epsilon)), 1+\epsilon)$-approximation for the ECC problem in $\mathbb{R}^d$. Thus, assuming we are allowed to expand the input balls by an arbitrarily small constant factor, we can obtain a solution with at most a corresponding constant times the number of balls used by the optimal solution. Without expansion, the best known approximation guarantee for $d \geq 3$ is $O(\log n)$, even without capacity constraints.

Both results are obtained via a unified scheme for rounding the natural LP relaxation for the problem. This scheme, which is instantiated in different ways to obtain the two results, may be of independent interest for obtaining similar results for related capacitated covering problems. Though the LP rounding technique is a standard tool that has been used in the literature of the capacitated problems, our actual rounding scheme is different from the existing ones. In fact, the standard rounding scheme for facility location, for example the one in [23], is not useful for our problems, as there a point can be assigned to any facility. But in our case, each point must be assigned to a ball that contains it (modulo constant factor expansion). This hard constraint makes the covering problems more complicated to deal with.

When the input balls have the same radius, it is easier to obtain the above guarantees for the MCC and and the ECC using known results or techniques. For the MCC, this (in fact, even a $(1, O(1))$-approximation) follows from the results for capacitated k-center [5, 21, 13, 2]. This is because of the connection between Capacitated k-center and MCC as pointed out before. The novelty in our work lies in handling the challenging scenario where the input balls have widely different radii. For geometric optimization problems, inputs with objects at multiple scales are often more difficult to handle than inputs with objects at the same scale.

As a byproduct of the rounding schemes we develop, the bicriteria approximations can be extended to a more general capacity model. In this model, the capacities of the balls are not necessarily the same. In particular, suppose ball B_i has capacity U_i and radius r_i. Then for any two balls $B_i, B_j \in \mathcal{B}$, our model assumes that the following holds: $r_i > r_j \implies U_i \geq U_j$. We refer to this capacity model as the *monotonic* capacity model. We refer to the generalizations of the MCC and the ECC problems with the monotonic capacity model as the *Metric Monotonic Capacitated Covering* (MMCC) problem and the *Euclidean Monotonic Capacitated Covering* (EMCC) problem, respectively. We note that the monotonicity assumption on the capacities is reasonable in many applications such as wireless networks – it might be economical to invest in capacity of an antenna to serve more clients, if it covers more area.

Hardness. We complement our algorithmic results with some hardness of approximation results that give a better understanding of the problems we consider. Firstly, we show that

for any constant $c > 1$, there exists a constant $\epsilon_c > 0$ such that it is NP-hard to obtain a $(1 + \epsilon_c, c)$-approximation for the MCC problem, even when the capacity of all balls is 3. This shows that it is not possible to obtain a $(1, c)$ approximation even for an arbitrarily large constant c. In the hardness construction, not all the balls in the hard instance have the same radius. This should be contrasted with the case where the radii of all balls are equal – in this case one can use the results from capacitated k-center (such as [2, 13]), to obtain a $(1, O(1))$-approximation.

It is natural to wonder if our algorithmic results can be extended to weighted versions of the problems. We derive hardness results that indicate that this is not possible. In particular, we show that for any constant $c \geq 1$, there exists a constant $c' > 0$, such that it is NP-hard to obtain a $(c' \log n, c)$-approximation for the weighted version of MMCC with a very simple weight function (a constant power of original radius).

We describe the natural LP relaxation for the MMCC problem in Section 2. We describe a unified rounding scheme in Section 3, and apply it in two different ways to obtain the algorithmic guarantees for MMCC and EMCC. We refer the reader to the full version of the paper for the results that cannot be accommodated here due to space constraints.

2 LP relaxation for MMCC

Recall that the input for the MMCC consists of a set P of points and a set $\mathcal{B}$ of balls in some metric space, along with an integer capacity $U_i > 0$ for ball $B_i \in \mathcal{B}$. We assume that for any two input balls $B_i, B_j \in \mathcal{B}$, it holds that $r_i > r_j \implies U_i \geq U_j$. The goal is to compute a minimum cardinality subset $\mathcal{B}' \subseteq \mathcal{B}$ for which each point in P can be assigned to a ball $\mathcal{B}'$ containing it in such a way that no more than U_i points are assigned to ball B_i. Let $d(p, q)$ denote the distance between two points p and q in the metric space. Let $B(c, r)$ denote the ball of radius r centered at point c. We let c_i and r_i denote the center and radius of ball $B_i \in \mathcal{B}$; thus, $B_i = B(c_i, r_i)$.

First we consider an integer programming formulation of MMCC. For each set $B_i \in \mathcal{B}$, let $y_i = 1$ if the ball B_i is selected in the solution, and 0 otherwise. Similarly, for each point $p_j \in X$ and each ball $B_i \in \mathcal{B}$, let the variable $x_{ij} = 1$ if p_j is assigned to B_i, and $x_{ij} = 0$ otherwise. We relax these integrality constraints, and state the corresponding linear program as follows:

$$\text{minimize} \quad \sum_{B_i \in \mathcal{B}} y_i \qquad \text{(MMCC-LP)}$$

$$\text{s.t.} \quad x_{ij} \leq y_i \qquad \forall p_j \in P,\ \forall B_i \in \mathcal{B} \qquad (1)$$

$$\sum_{p_j \in P} x_{ij} \leq y_i \cdot U_i \qquad \forall B_i \in \mathcal{B} \qquad (2)$$

$$\sum_{B_i \in \mathcal{B}} x_{ij} = 1 \qquad \forall p_j \in P \qquad (3)$$

$$x_{ij} = 0 \qquad \forall p_j \in P,\ \forall B_i \in \mathcal{B} \text{ such that } p_j \notin \mathcal{B}_i \qquad (4)$$

$$x_{ij} \geq 0 \qquad \forall p_j \in P,\ \forall B_i \in \mathcal{B} \qquad (5)$$

$$0 \leq y_i \leq 1 \qquad \forall B_i \in \mathcal{B} \qquad (6)$$

Subsequently, we will refer to an assignment (x, y) that is feasible or infeasible with respect to Constraints 1-6 as just a *solution*. The *cost* of the LP solution $\sigma = (x, y)$ (feasible or otherwise), denoted by $\text{cost}(\sigma)$, is defined as $\sum_{B_i \in \mathcal{B}} y_i$.

3 The algorithmic framework

In this section, we describe our framework for extracting an integral solution from a fractional solution to the above LP. The framework consists of two major steps – Preprocessing and the Main Rounding. The Main Rounding step is in turn divided into two smaller steps – Cluster Formation and Selection of Objects. For simplicity of exposition, we first describe the framework with respect to the MMCC problem as an algorithm and analyze the approximation factor achieved by this algorithm for MMCC. Later, we show how one or more steps of this algorithm can be modified to obtain the desired results for the EMCC.

3.1 The algorithm for the MMCC problem

Before we describe the algorithm we introduce some definitions and notation which will heavily be used throughout this section. For point $p_j \in P$ and ball $B_i \in \mathcal{B}$, we refer to x_{ij} as the *flow* from B_i to p_j; if $x_{ij} > 0$, then we say that the ball B_i *serves* the point p_j. Each ball $B_i \in \mathcal{B}$ can be imagined as a source of at most $y_i \cdot U_i$ units of flow, which it distributes to some points in P.

We now define an important operation, called *rerouting of flow*. "Rerouting of flow for a set $P' \subseteq P$ of points from a set of balls $\mathcal{B}'$ to a ball $B_k \notin \mathcal{B}'$" means obtaining a new solution $(\hat{x}, \hat{y})$ from the current solution (x, y) in the following way: (a) For all points $p_j \in P'$, $\hat{x}_{kj} = x_{kj} + \sum_{B_i \in \mathcal{B}'} x_{ij}$; (b) for all points $p_j \in P'$ and balls $B_i \in \mathcal{B}'$, $\hat{x}_{ij} = 0$; (c) the other $\hat{x}_{ij}$ variables are the same as the corresponding x_{ij} variables. The relevant $\hat{y}_i$ variables may also be modified depending on the context where this operation is used.

Let $0 < \alpha \leq \frac{1}{2}$ be a parameter to be fixed later. A ball $B_i \in \mathcal{B}$ is *heavy* if the corresponding $y_i = 1$, and *light*, if $0 < y_i \leq \alpha$. Corresponding to a feasible LP solution (x, y), let $\mathcal{H} = \{B_i \in \mathcal{B} \mid y_i = 1\}$ denote the set of heavy balls, and $\mathcal{L} = \{B_i \in \mathcal{B} \mid 0 < y_i \leq \alpha\}$ denote the set of light balls. We emphasize that the set $\mathcal{L}$ of light and $\mathcal{H}$ of heavy balls are defined w.r.t. an LP solution; however, the reference to the LP solution may be omitted when it is clear from the context.

Now we move on towards the description of the algorithm. The algorithm, given a feasible fractional solution $\sigma = (x, y)$, rounds σ to a solution $\hat{\sigma} = (\hat{x}, \hat{y})$ such that $\hat{y}$ is integral, and the cost of $\hat{\sigma}$ is within a constant factor of the cost of σ. The $\hat{x}$ variables are non-negative but may be fractional. Furthermore, each point receives unit flow from the balls that are chosen (y values are 1), and the amount of flow each chosen ball sends is bounded by its capacity. Notably, no point gets any non-zero amount of flow from a ball that is not chosen (y value is 0). Moreover, for any ball B_i and any $p_j \in P$, if B_i serves p_j, then $d(c_i, p_j)$ is at most a constant times r_i. We expand each ball by a constant factor so that it contains all the points it serves.

We note that in $\hat{\sigma}$ points might receive fractional amount of flow from the chosen balls. However, since the capacity of each ball is integral, we can find, using a textbook argument for integrality of flow, another solution with the same set of chosen balls, such that the new solution satisfies all the properties of $\hat{\sigma}$ and the additional property, that for each point p, there is a single chosen ball that sends one unit of flow to p [11]. Thus, choosing an optimal LP solution as the input $\sigma = (x, y)$ of the rounding algorithm yields a constant approximation for MMCC by expanding each ball by at most a constant factor.

Our LP rounding algorithm consists of two steps. The first step is a preprocessing step where we construct a fractional LP solution $\overline{\sigma} = (\overline{x}, \overline{y})$ from σ, such that each ball in $\overline{\sigma}$ is either heavy or light, and for each point $p_j \in P$, the amount of flow that p_j can potentially receive from the light balls is at most α. The latter property will be heavily exploited in the

next step. The second step is the core step of the algorithm where we round $\overline{\sigma}$ to the desired integral solution.

We note that throughout the algorithm, for any intermediate LP solution that we consider, we maintain the following two invariants: (i) Each ball B_i sends at most U_i units of flow to the points, and (ii) Each point receives exactly one unit of flow from the balls. With respect to a solution $\sigma = (x, y)$, we define the *available capacity* of a ball $B_i \in \mathcal{B}$, denoted $\text{AvCap}(B_i)$, to be $U_i - \sum_{p_j \in P} x_{ij}$. We now describe the preprocessing step.

3.1.1 The preprocessing step

▶ **Lemma 1.** *Given a feasible LP solution $\sigma = (x, y)$, and a parameter $0 < \alpha \leq \frac{1}{2}$, there exists a polynomial time algorithm to obtain another LP solution $\overline{\sigma} = (\overline{x}, \overline{y})$ that satisfies Constraints 1-6 except 4 of MMCC-LP. Additionally, $\overline{\sigma}$ satisfies the following properties.*

1. *Any ball $B_i \in \mathcal{B}$ with non-zero $\overline{y_i}$ is either heavy ($\overline{y_i} = 1$) or light ($0 < \overline{y_i} \leq \alpha$).*
2. *For each point $p_j \in P$, we have that*

$$\sum_{B_i \in \mathcal{L}: \overline{x}_{ij} > 0} \overline{y_i} \leq \alpha, \tag{7}$$

   where $\mathcal{L}$ is the set of light balls with respect to $\overline{\sigma}$.
3. *For any heavy ball B_i, and any point $p_j \in P$ served by B_i, $d(c_i, p_j) \leq 3r_i$.*
4. *For any light ball B_i, and any point $p_j \in P$ served by B_i, $d(c_i, p_j) \leq r_i$.*
5. $\text{cost}(\overline{\sigma}) \leq \frac{1}{\alpha}\text{cost}(\sigma)$.

Proof. The algorithm starts off by initializing $\overline{\sigma}$ to σ. While there is a violation of Inequality 7, we perform the following steps.

1. We pick an arbitrary point $p_j \in P$, for which Inequality 7 is not met. Let $\mathcal{L}_j$ be a subset of light balls serving p_j such that $\alpha < \sum_{B_i \in \mathcal{L}_j} \overline{y}_i \leq 2\alpha$. Note that such a set $\mathcal{L}_j$ always exists because the $\overline{y}_i$ variables corresponding to light balls are at most $\alpha \leq \frac{1}{2}$. Let B_k be a ball with the largest radius from the set $\mathcal{L}_j$. (If there is more than one ball with the largest radius, we consider one having the largest capacity among those. Throughout the paper we follow this convention.) Since $r_k \geq r_m$ for all other balls $B_m \in \mathcal{L}_j$, we have, by the *monotonicity* assumption, that $U_k \geq U_m$.
2. We set $\overline{y}_k \leftarrow \sum_{B_i \in \mathcal{L}_j} \overline{y}_i$, and $\overline{y}_m \leftarrow 0$ for all $B_m \in \mathcal{L}_j \setminus \{B_k\}$. Note that $\overline{y}_k \leq 2\alpha \leq 1$. Let $A = \{p_t \in P \mid \overline{x}_{it} > 0 \text{ for some } B_i \in \mathcal{L}_j \setminus \{B_k\}\}$ be the set of "affected" points. We reroute the flow for all the affected points in A from $\mathcal{L}_j \setminus \{B_k\}$ to the ball B_k. Since $U_k \geq U_m$ for all other balls $B_m \in \mathcal{L}_j$, B_k has enough available capacity to "satisfy" all "affected" points. In $\overline{\sigma}$, all other $\overline{x}_{ij}$ and $\overline{y_i}$ variables remain same as before. (*Note*: Since B_k had the largest radius from the set $\mathcal{L}_j$, all the points in A are within distance $3r_k$ from its center c_k, as seen using the triangle inequality. Also, since $\overline{y}_k > \alpha$, B_k is no longer a light ball.)

Finally, for all balls B_i such that $\overline{y}_i > \alpha$, we set $\overline{y}_i = 1$, making them heavy. Thus $\text{cost}(\overline{\sigma})$ is at most $\frac{1}{\alpha}$ times $\text{cost}(\sigma)$, and $\overline{\sigma}$ satisfies all the conditions stated in the lemma. ◀

Remark. As a byproduct of Lemma 1, we get a simple $(4, 3)$-approximation algorithm for the *soft* capacitated version of our problem that allows making multiple copies of the input balls.

3.1.2 The main rounding step

The main rounding step can be logically divided into two stages. The first stage, *Cluster Formation*, is the crucial stage of the algorithm. Note that there can be many light balls in the preprocessed solution. Including all these balls in the final solution may incur a huge cost. Thus we use a careful strategy based on flow rerouting to select a small number of balls. The idea is to use the capacity of a selected light ball to reroute as much flow as possible from other intersecting balls. This in turn frees up some capacity at those balls. The available capacity of each heavy ball is used, when possible, to reroute *all* the flow from some light ball intersecting it; this light ball is then added to a cluster centered around the heavy ball. Notably, for each cluster, the heavy ball is the only ball in it that actually serves some points, as we have rerouted flow from the other balls in the cluster to the heavy ball. In the second stage, referred to as *Selection of Objects*, we select exactly one ball (in particular, a largest ball) from each cluster as part of the final solution, and reroute the flow from the heavy ball to this ball, and expand it by the required amount. Together, these two stages ensure that we do not end up choosing many light balls.

We now describe the two stages in detail. Recall that any ball in the preprocessed solution is either heavy or light. Also $\mathcal{L}$ denotes the set of light balls and $\mathcal{H}$ the set of heavy balls. Note that any heavy ball B_i may serve a point p_j which is at a distance $3r_i$ from c_i. We expand each heavy ball by a factor of 3 so that B_i can contain all points it serves.

1. **Cluster formation.** In this stage, each light ball, will be added to either a set $\mathcal{O}$ (that will eventually be part of the final solution), or a cluster corresponding to some heavy ball. Till the very end of this stage, the sets of heavy and light balls remain unchanged. The set $\mathcal{O}$ is initialized to $\emptyset$. For each heavy ball B_i, we initialize the cluster of B_i, denoted by cluster(B_i) to $\{B_i\}$. We say a ball is clustered if it is added to a cluster.
   At any point, let Λ denote the set consisting of each light ball that is (a) not in $\mathcal{O}$, and (b) not yet clustered. While the set Λ is non-empty, we perform the following steps.
   a. While there is a heavy ball B_i and a light ball $B_t \in \Lambda$ such that (1) B_t intersects B_i; and (2) AvCap(B_i) is at least the flow $\sum_{p_j \in P} \overline{x}_{tj}$ out of B_t:
      1. For all the points served by B_t, we reroute the flow from B_t to B_i.
      2. We add B_t to cluster(B_i).

      After the execution of this *while* loop, if the set Λ becomes empty, we stop and proceed to the *Selection of Objects* stage. Otherwise, we proceed to the following.
   b. For any ball $B_j \in \Lambda$, let $\mathcal{A}_j$ denote the set of points currently being served by B_j. Also, for $B_j \in \Lambda$, let $k_j = \min\{U_j, |\mathcal{A}_j|\}$, i.e. k_j denotes the minimum of its capacity, and the number of points that it currently serves. We select the ball $B_t \in \Lambda$ with the maximum value of k_j, and add it to the set $\mathcal{O}$.
   c. Since we have added B_t to the set $\mathcal{O}$ that will be among the selected balls, we use the available capacity at B_t to reroute flow to it. This is done based on the following three cases depending on the value of k_t.

      1. $k_t = |\mathcal{A}_t| \leq U_t$. In this case, for each point p_l in B_t that gets served by B_t, we reroute the flow of p_l from $\mathcal{B} \setminus \mathcal{O}$ to B_t. Note that after the rerouting, p_l is no longer being served by a ball in Λ. The rerouting increases the available capacity of other balls intersecting B_t. In particular, for each $B_i \in \mathcal{H}$, AvCap(B_i) increases by $\sum_{p_l : B_t \text{ serves } p_l} \overline{x}_{il}$.

      2. $k_t = U_t < |\mathcal{A}_t|$, but $k_t = U_t > 1$. Observe that the flow out of ball B_t is $\sum_{p_j \in \mathcal{A}_t} x_{tj} \leq \alpha U_t$; thus AvCap$(B_t) \geq (1-\alpha)U_t = (1-\alpha)k_t$.

      In this case, we select a point $p_j \in \mathcal{A}_t$ arbitrarily, and reroute the flow of p_j from

$\mathcal{B} \setminus \mathcal{O}$ to B_t. This will increase the available capacity of other balls in $\mathcal{B} \setminus \mathcal{O}$ that were serving p_j. Also note that p_j is no longer being served by a ball in Λ.

We repeat the above flow rerouting process for other points of $\mathcal{A}_t$ until we encounter a point p_l such that rerouting the flow of p_l from $\mathcal{B} \setminus \mathcal{O}$ to B_t violates the capacity of B_t. Thus the flow assignment of p_l remains unchanged. Note that we can reroute the flow of at least $\lfloor (1-\alpha)k_t \rfloor = \lfloor (1-\alpha)U_t \rfloor \geq 1$ points of $\mathcal{A}_t$ in this manner, since $U_t > 1$ and $\alpha \leq 1/2$.

3. $k_t = U_t = 1 < |\mathcal{A}_t|$. Note that B_t has used $\sum_{p_j \in \mathcal{A}_t} x_{tj} \leq \alpha U_t = \alpha$ capacity. In this case, we pick a point $p_j \in \mathcal{A}_t$ arbitrarily, and then perform the following two steps:

- **(i)** Reroute the flow of p_j from Λ to B_t; after this, p_j is no longer being served by a ball in Λ. Note that in this step, we reroute at most α amount of flow. Therefore, at this point we have $\text{AvCap}(B_t) \geq 1 - 2\alpha$. Let f be the amount of flow p_j receives from the balls in $\mathcal{O}$.
- **(ii)** Then we reroute $\min\{\text{AvCap}(B_t), 1-f\}$ amount of flow of p_j from the set $\mathcal{H}$ to B_t.

When the loop terminates, we have that each light ball is either in $\mathcal{O}$ or clustered. We set $\overline{y}_i \leftarrow 1$ for each ball $B_i \in \mathcal{O}$, thus making it heavy. For convenience, we also set $\text{cluster}(B_i) = \{B_i\}$ for each $B_i \in \mathcal{O}$. Note that, throughout the algorithm we ensure that, if a point $p_j \in P$ is currently served by a ball $B_i \in \Lambda$, then the amount of flow p_j receives from any ball $B_{i'} \in \mathcal{O} \cup \Lambda$ is the same as that in the preprocessed solution, i.e., the flow assignment of p_j w.r.t. the balls in $\mathcal{O} \cup \Lambda$ remains unchanged.

2. **Selection of objects.** At the start of this stage, we have a collection of clusters, each centered around a heavy ball, such that the light balls in each cluster intersect the heavy ball. We are going to pick exactly one ball from each cluster and add it to a set $\mathcal{C}$. Let $\mathcal{C} = \emptyset$ initially. For each heavy ball B_i, we consider $\text{cluster}(B_i)$ and perform the following steps.
   - **a.** If $\text{cluster}(B_i)$ consists of only the heavy ball, we add B_i to $\mathcal{C}$.
   - **b.** Otherwise, let B_j be a largest ball in $\text{cluster}(B_i)$. If $B_j = B_i$, then we expand it by a factor of 3. Otherwise, B_j is a light ball intersecting with B_i, in which case we expand it by a factor of 5. In this case, we also reroute the flow from the heavy ball to the selected ball B_j. Note that since we always choose a largest ball in the cluster, its capacity is at least that of the heavy ball, because of the *monotonicity* assumption. We add B_j to $\mathcal{C}$, and we set $\overline{y}_s \leftarrow 0$ for any other ball B_s in the cluster.

   After processing the clusters, we set $\overline{y}_t \leftarrow 1$ for each ball $B_t \in \mathcal{C}$. Finally, we return the current set of heavy balls (i.e., $\mathcal{C}$) as the set of selected balls. Note that the flow out of each such ball is at most its capacity, and each point receives one unit of flow from the (possibly expanded) balls that contain it. As mentioned earlier, this can be converted into an integral flow.

3.1.3 The analysis of the rounding algorithm

Let OPT be the cost of an optimal solution. We establish a bound on the number of balls our algorithm outputs by bounding the size of the set $\mathcal{C}$. Then we conclude by showing that any input ball that is part of our solution expands by at most a constant factor to cover the points it serves.

For notational convenience, we refer to the solution $\overline{\sigma} = (\overline{x}, \overline{y})$ at hand after preprocessing, as $\sigma = (x, y)$. Now we bound the size of the set $\mathcal{O}$ computed during Cluster Formation. The basic idea is that each light ball added to $\mathcal{O}$ creates significant available capacity in the heavy balls. Furthermore, whenever there is enough available capacity, a heavy ball

clusters intersecting light balls, thus preventing them from being added to $\mathcal{O}$. The actual argument is more intricate because we need to work with a notion of y-accumulation, a proxy for available capacity. The way the light balls are picked for addition to $\mathcal{O}$ plays a crucial role in the argument.

Let $\mathcal{H}_1$ (resp. $\mathcal{L}_1$) be the set of heavy (resp. light) balls after preprocessing, and I be the total number of iterations in the Cluster Formation stage. Also let L_j be the light ball selected (i.e. added to $\mathcal{O}$) in iteration j for $1 \leq j \leq I$. Now, L_t maximizes k_j amongst all balls from Λ in iteration t (Recall that k_j was defined as the minimum of the number of points being served by L_j, and its capacity). Note that $k_1 \geq k_2 \geq \cdots \geq k_I$. For any $B_i \in \mathcal{H}_1$, denote by $F(L_t, B_i)$, the total amount of flow rerouted in iteration t from B_i to L_t corresponding to the points B_i serves. This is the same as the increase in $\mathrm{AvCap}(B_i)$ when L_t is added to $\mathcal{O}$. Correspondingly, we define $Y(L_t, B_i)$, the "y-credit contributed by L_t to B_i", to be $\frac{F(L_t,B_i)}{k_t}$. Now, the increase in available capacity over all balls in $\mathcal{H}_1$ is $F_t = \sum_{B_i \in \mathcal{H}_1} F(L_t, B_i)$. The approximation guarantee of the algorithm depends crucially on the following simple lemma, which states that in each iteration we make "sufficiently large" amount of flow available for the set of heavy balls.

▶ **Lemma 2.** *Consider a ball $B_t \in \mathcal{O}$ processed in the* Cluster Formation *stage, step c. For $0 < \alpha \leq 3/8$, $F_t \geq \frac{1}{5}k_t$.*

Proof. The algorithm ensures that the flow assignment of each point in $\mathcal{A}_t$ w.r.t. the balls in $\mathcal{O} \cup \Lambda$ is the same as that in the preprocessed solution. Thus by property 2 of Lemma 1, each such point gets at most α amount of flow from the balls in $\mathcal{O} \cup \Lambda$. Now there are three cases corresponding to the three substeps of step c.

1. $k_t = |\mathcal{A}_t| \leq U_t$. For each point in $\mathcal{A}_t$, at most α amount of flow comes from the balls in $\mathcal{O} \cup \Lambda$. So the remainder is rerouted from the balls in $\mathcal{H}_1$ resulting in a contribution of at least $1 - \alpha$ towards F_t. Therefore, we get that $F_t \geq (1-\alpha)k_t \geq \frac{1}{5}k_t$, since $0 < \alpha \leq 3/8$.
2. $1 < k_t = U_t < |\mathcal{A}_t|$. It is possible to reroute the flow of at least $\lfloor (1-\alpha)U_t \rfloor = \lfloor (1-\alpha)k_t \rfloor$ points of $\mathcal{A}_t$ from $\mathcal{B} \setminus \mathcal{O}$ to B_t. Therefore, we get that $F_t \geq (1-\alpha)\lfloor (1-\alpha)k_t \rfloor$. When $k_t > 1$, the previous quantity is at least $\frac{1}{5}k_t$, again by using the fact that $0 < \alpha \leq 3/8$.
3. When $1 = k_t = U_t < |\mathcal{A}_t|$, $F_t \geq (1-2\alpha) \geq \frac{1}{5}k_t$, as $0 < \alpha \leq 3/8$. ◀

At any moment in the Cluster Formation stage, for any ball $B_i \in \mathcal{H}_1$, define its y-accumulation as

$$\widetilde{y}(B_i) = \Bigg(\sum_{L_t \in \mathcal{O}} Y(L_t, B_i) \Bigg) - \Bigg(\sum_{B_j \in \mathcal{L} \cap \text{ cluster}(B_i)} y_j \Bigg).$$

The idea is that B_i gets y-credit when a light ball is added to $\mathcal{O}$, and loses y-credit when it adds a light ball to cluster(B_i); thus, $\widetilde{y}(B_i)$, a proxy for the available capacity of B_i, indicates the "remaining" y-credit. The next lemma gives a relation between the y-accumulation of B_i and its available capacity.

▶ **Lemma 3.** *Fix a heavy ball $B_i \in \mathcal{H}_1$, and an integer $1 \leq t \leq I$. Suppose that $L_1, L_2, \cdots, L_t$ have been added to $\mathcal{O}$. Then AvCap$(B_i) \geq \widetilde{y}(B_i) \cdot k_t$.*

Proof. The proof is by induction on t. For this proof, we abbreviate $\mathrm{AvCap}(B_i)$ by A_i. In the first iteration, just after adding L_1, $A_i \geq F(L_1, B_i) = Y(L_1, B_i) \cdot k_1 \geq \widetilde{y}(B_i) \cdot k_1$.

Assume inductively that we have added balls $L_1, \cdots, L_{t-1}$ to the set $\mathcal{O}$, and that just after adding L_{t-1}, the claim is true. That is, if $\widetilde{y}(B_i)$ and A_i are, respectively, the y-accumulation and the available capacity of B_i just after adding L_{t-1}, then $A_i \geq \widetilde{y}(B_i) \cdot k_{t-1}$.

Consider the iteration t. At step (a) of Cluster Formation, B_i uses up some of its available capacity to add 0 or more balls to cluster(B_i), after which at step (b) we add L_t to $\mathcal{O}$. Suppose that at step (a), one or more balls are added to cluster(B_i). Let B_j be the first such ball, and let k and C_1 be the number of points B_j serves and the capacity of B_j, respectively. Then the amount of capacity used by B_j is at most

$$\min\{C_1 \cdot y_j, k \cdot y_j\} = \min\{C_1, k\} \cdot y_j \leq k_{t-1} \cdot y_j$$

where the last inequality follows because of the order in which we add balls to $\mathcal{O}$. Now, after adding B_j to cluster(B_i), the new y-accumulation becomes $\widetilde{y}(B_i)' = \widetilde{y}(B_i) - y_j$. As for the available capacity,

$$A_i' \geq A_i - k_{t-1} \cdot y_j \geq (\widetilde{y}(B_i) \cdot k_{t-1}) - k_{t-1} \cdot y_j \geq (\widetilde{y}(B_i) - y_j) \cdot k_{t-1} = \widetilde{y}(B_i)' \cdot k_{t-1}$$

Therefore, the claim is true after addition of the first ball B_j. Note that B_i may add multiple balls to cluster(B_i), and the preceding argument would work after each such addition.

Now consider the moment when L_t is added to $\mathcal{O}$. Let $\widetilde{y}(B_i)$ denote the y-accumulation just before this. Now, the new y-accumulation of B_i becomes $\widetilde{y}(B_i)' = \widetilde{y}(B_i) + Y(L_t, B_i)$. If $\widetilde{y}(B_i) \leq 0$, then the new available capacity is

$$A_i' \geq F(L_t, B_i) = Y(L_t, B_i) \cdot k_t \geq \widetilde{y}(B_i)' \cdot k_t.$$

If $\widetilde{y}(B_i) > 0$, the new available capacity, using the inductive hypothesis, is

$$A_i' \geq \widetilde{y}(B_i) \cdot k_{t-1} + Y(L_t, B_i) \cdot k_t \geq (\widetilde{y}(B_i) + Y(L_t, B_i)) \cdot k_t = \widetilde{y}(B_i)' \cdot k_t$$

where, in the second inequality we use $k_t \leq k_{t-1}$. ◀

Now, in the next lemma, we show that any ball $B_i \in \mathcal{H}_1$ cannot have "too-much" y-accumulation at any moment during Cluster Formation.

▶ **Lemma 4.** *At any moment in the Cluster Formation stage, for any ball $B_i \in \mathcal{H}_1$, we have that $\widetilde{y}(B_i) \leq 1 + \alpha$.*

Proof. The proof is by contradiction. Let $B_i \in \mathcal{H}_1$ be the first ball that violates the condition. As $\widetilde{y}(B_i)$ increases only due to addition of a light ball to set $\mathcal{O}$, suppose L_t was the ball whose addition to $\mathcal{O}$ resulted in the violation.

Let $\widetilde{y}(B_i)$ and $\widetilde{y}(B_i)' = \widetilde{y}(B_i) + Y(L_t, B_i)$ be the y-accumulations of B_i just before and just after the addition of L_t. Because of the assumption, $\widetilde{y}(B_i) \leq 1 + \alpha$. So the increase in the y-accumulation of B_i must be because $Y(L_t, B_i) > 0$. Thus, L_t intersects B_i. However, $Y(L_t, B_i) \leq 1$ by definition. Therefore, we have $\widetilde{y}(B_i) > \alpha$.

Now, by Lemma 3, just before addition of L_t, $\text{AvCap}(B_i) \geq \widetilde{y}(B_i) \cdot k_{t-1} > \alpha \cdot k_{t-1} \geq \alpha \cdot k_t$, as $k_t \leq k_{t-1}$. However, L_t is a light ball, and so the total flow out of L_t is at most αk_t. Therefore, the available capacity of B_i is large enough that we can add L_t to cluster(B_i), instead of to the set $\mathcal{O}$, which is a contradiction. ◀

▶ **Lemma 5.** *At the end of Cluster Formation stage, we have $|\mathcal{O}| \leq 5 \cdot \left((1+\alpha) \cdot |\mathcal{H}_1| + \sum_{B_j \in \mathcal{L}_1} y_j\right)$, where $0 < \alpha \leq 3/8$.*

Proof. At the end of Cluster Formation stage,

$$\begin{aligned}\sum_{B_i\in\mathcal{H}_1}\widetilde{y}(B_i) &\geq \sum_{\substack{B_i\in\mathcal{H}_1\\1\leq t\leq I}} Y(L_t,B_i) - \sum_{B_i\in\mathcal{H}_1}\sum_{B_j\in\text{ cluster}(B_i)} y_j\\ &\geq \sum_{1\leq t\leq I}(F_t/k_t) - \sum_{B_j\in\mathcal{L}_1} y_j\\ &\qquad\qquad (\because F_t = \textstyle\sum_{B_i\in\mathcal{H}_1} F(L_t,B_i) = k_t\cdot\sum_{B_i\in\mathcal{H}_1} Y(L_t,B_i))\\ &\geq \frac{1}{5}\cdot|\mathcal{O}| - \sum_{B_j\in\mathcal{L}_1} y_j \qquad (8)\end{aligned}$$

Where we used Observation 2 to get the last inequality.
Now, adding the inequality of Lemma 4 over all $B_i\in\mathcal{H}_1$, we have that $\sum_{B_i\in\mathcal{H}_1}\widetilde{y}(B_i)\leq(1+\alpha)\cdot|\mathcal{H}_1|$. Combining this with (8) yields the desired inequality. ◀

▶ **Lemma 6.** *The cost of the solution returned by the algorithm is at most* 21 *times the cost of an optimal solution.*

Proof. Let $\sigma=(x,y)$ be the preprocessed LP solution. Now, the total number of balls in the solution is $|\mathcal{O}|+|\mathcal{H}_1|$. Using Lemma 5,

$$\begin{aligned}|\mathcal{O}|+|\mathcal{H}_1| &\leq 5\cdot\left((1+\alpha)\cdot|\mathcal{H}_1| + \sum_{B_j\in\mathcal{L}_1} y_j\right) + |\mathcal{H}_1|\\ &\leq (6+5\alpha)\left(\sum_{B_j\in\mathcal{H}_1} y_j + \sum_{B_j\in\mathcal{L}_1} y_j\right)\\ &\leq (6+5\alpha)\cdot\text{cost}(\sigma)\\ &\leq \left(\frac{6+5\alpha}{\alpha}\right)\cdot OPT = 21\cdot OPT \qquad (\text{by setting } \alpha = 3/8)\end{aligned}$$

◀

▶ **Lemma 7.** *In the algorithm each input ball is expanded by at most a factor of* 9*.*

Proof. Recall that when a light ball becomes heavy in the preprocessing step, it is expanded by a factor of 3. Therefore after the preprocessing step, any heavy ball in a solution may be an expanded or unexpanded ball.

Now, consider the selection of the balls in the second stage. If a cluster consists of only a heavy ball, then it does not expand any further. Since it might be an expanded light ball, the total expansion factor is at most 3.

Otherwise, for a fixed cluster, let r_l and r_h be the radius of the largest light ball and the heavy ball, respectively. If $r_l\geq r_h$, then the overall expansion factor is 5. Otherwise, if $r_l<r_h$, then the heavy ball is chosen, and it is expanded by a factor of at most 3. Now as the heavy ball might already be expanded by a factor of 3 during the preprocessing step, here the overall expansion factor is 9. ◀

If the capacities of all balls are equal, then one can improve the expansion factor to 6.47 by using an alternative procedure to the *Selection of Balls* stage. Lastly, from Lemmas 6 and 7, we get the following theorem.

▶ **Theorem 8.** *There is a polynomial time* $(21,9)$*-approximation algorithm for the MMCC problem.*

3.2 The algorithm for the EMCC problem

Overview of the algorithm. For the EMCC problem, we can exploit the structure of $\mathbb{R}^d$ to restrict the expansion factor of the balls to at most $(1+\epsilon)$, while paying in terms of the cost of the solution. In the following, we give an overview of how to adapt the stages of the framework for obtaining this result. Note that in each iteration of the preprocessing stage for MMCC, we consider a point p_j and a cluster $\mathcal{L}_j$ of light balls. We select a largest ball from this set and reroute the flow of other balls in $\mathcal{L}_j$ to this ball. However, to ensure that the selected ball contains all the points it serves we need to expand this ball by a factor of 3. For the EMCC problem, for the cluster $\mathcal{L}_j$, we consider the bounding hypercube whose side is at most a constant times the maximum radius of the balls from $\mathcal{L}_j$, and subdivide it into multiple cells. The granularity of the cells is carefully chosen to ensure that (1) Selecting a maximum radius ball among the balls whose centers are lying in that cell, and expanding it by $(1+\epsilon)$ factor is enough to cover all such balls, and (2) The total number of cells is $poly(1/\epsilon)$. From each cell we select a maximum radius ball, expand it by $(1+\epsilon)$ factor, and reroute the flow from the other balls in that cell to it. It follows that the cost of the preprocessed solution goes up by at most a $poly(1/\epsilon)$ factor. The Cluster Formation stage for the EMCC problem is exactly the same as that for the MMCC problem. Note that, in the Selection of Balls stage for MMCC, we select only one ball per cluster and expand it by $O(1)$ factor to cover all the balls in the cluster. But in case of EMCC, we want to restrict the expansion factor of the balls to at most $(1+\epsilon)$. Hence we select $poly(1/\epsilon)$ number of balls per cluster in a way so that the selected balls when expanded by a factor of $(1+\epsilon)$ can cover all the balls in the cluster. The ideas are similar to the ones in the Preprocessing stage, however one needs to be more careful to handle some technicalities that arise. We summarize the result for the EMCC problem in the following theorem.

▶ **Theorem 9.** *There is a polynomial time* $(O(\epsilon^{-4d}\log(1/\epsilon)), 1+\epsilon)$*-approximation algorithm for the EMCC problem in* $\mathbb{R}^d$*, for any* $\epsilon > 0$.

References

1 Anshul Aggarwal, Venkatesan T. Chakaravarthy, Neelima Gupta, Yogish Sabharwal, Sachin Sharma, and Sonika Thakral. Replica placement on bounded treewidth graphs. In *Algorithms and Data Structures - 15th International Symposium, WADS 2017, St. John's, NL, Canada, July 31 - August 2, 2017, Proceedings*, pages 13–24, 2017. doi:10.1007/978-3-319-62127-2_2.

2 Hyung-Chan An, Aditya Bhaskara, Chandra Chekuri, Shalmoli Gupta, Vivek Madan, and Ola Svensson. Centrality of trees for capacitated k-center. *Math. Program.*, 154(1-2):29–53, 2015. doi:10.1007/s10107-014-0857-y.

3 Hyung-Chan An, Mohit Singh, and Ola Svensson. Lp-based algorithms for capacitated facility location. In *FOCS*, pages 256–265, 2014.

4 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.

5 Judit Bar-Ilan, Guy Kortsarz, and David Peleg. How to allocate network centers. *J. Algorithms*, 15(3):385–415, 1993. URL: http://dblp.uni-trier.de/db/journals/jal/jal15.html#Bar-IlanKP93.

6 Amariah Becker. Capacitated dominating set on planar graphs. In *Approximation and Online Algorithms - 15th International Workshop, WAOA 2017, Vienna, Austria, September 7-8*, 2017.

7 Piotr Berman, Marek Karpinski, and Andrzej Lingas. Exact and approximation algorithms for geometric and capacitated set cover problems. *Algorithmica*, 64(2):295–310, 2012.

8 Prosenjit Bose, Paz Carmi, Mirela Damian, Robin Y. Flatland, Matthew J. Katz, and Anil Maheshwari. Switching to directional antennas with constant increase in radius and hop distance. *Algorithmica*, 69(2):397–409, 2014.

9 Hervé Brönnimann and Michael T. Goodrich. Almost optimal set covers in finite vc-dimension. *Discrete & Computational Geometry*, 14(4):463–479, 1995. doi:10.1007/BF02570718.

10 Timothy M. Chan, Elyot Grant, Jochen Könemann, and Malcolm Sharpe. Weighted capacitated, priority, and geometric set cover via improved quasi-uniform sampling. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 1576–1585, 2012. URL: https://dl.acm.org/citation.cfm?id=2095241.

11 Julia Chuzhoy and Joseph Naor. Covering problems with hard capacities. *SIAM J. Comput.*, 36(2):498–515, 2006.

12 Kenneth L. Clarkson and Kasturi R. Varadarajan. Improved approximation algorithms for geometric set cover. *Discrete & Computational Geometry*, 37(1):43–58, 2007.

13 Marek Cygan, MohammadTaghi Hajiaghayi, and Samir Khuller. LP rounding for k-centers with non-uniform hard capacities. In *FOCS*, pages 273–282, 2012.

14 Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 624–633, 2014. doi:10.1145/2591796.2591884.

15 Rajiv Gandhi, Eran Halperin, Samir Khuller, Guy Kortsarz, and Srinivasan Aravind. An improved approximation algorithm for vertex cover with hard capacities. *J. Comput. Syst. Sci.*, 72(1):16–33, 2006.

16 Taha Ghasemi and Mohammadreza Razzazi. A PTAS for the cardinality constrained covering with unit balls. *Theor. Comput. Sci.*, 527:50–60, 2014.

17 Sathish Govindarajan, Rajiv Raman, Saurabh Ray, and Aniket Basu Roy. Packing and covering with non-piercing regions. In *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*, pages 47:1–47:17, 2016. doi:10.4230/LIPIcs.ESA.2016.47.

18 Sariel Har-Peled and Mira Lee. Weighted geometric set cover problems revisited. *JoCG*, 3(1):65–85, 2012.

19 Dorit S. Hochbaum and Wolfgang Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *J. ACM*, 32(1):130–136, 1985.

20 Mong-Jen Kao. Iterative partial rounding for vertex cover with hard capacities. In *SODA*, pages 2638–2653, 2017.

21 Samir Khuller and Yoram J. Sussmann. The capacitated K-center problem. *SIAM J. Discrete Math.*, 13(3):403–418, 2000.

22 Nissan Lev-Tov and David Peleg. Polynomial time approximation schemes for base station coverage with minimum total radii. *Computer Networks*, 47(4):489–501, 2005.

23 Retsef Levi, David B. Shmoys, and Chaitanya Swamy. Lp-based approximation algorithms for capacitated facility location. *Math. Program.*, 131(1-2):365–379, 2012. doi:10.1007/s10107-010-0380-8.

24 Shi Li. On uniform capacitated k-median beyond the natural LP relaxation. In *SODA*, pages 696–707, 2015.

25 Robert Lupton, F. Miller Maley, and Neal E. Young. Data collection for the sloan digital sky survey - A network-flow heuristic. *J. Algorithms*, 27(2):339–356, 1998. doi:10.1006/jagm.1997.0922.

26 Nabil H. Mustafa and Saurabh Ray. Improved results on geometric hitting set problems. *Discrete & Computational Geometry*, 44(4):883–895, 2010. doi:10.1007/s00454-010-9285-9.

27 Kasturi R. Varadarajan. Weighted geometric set cover via quasi-uniform sampling. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 641–648, 2010. doi:10.1145/1806689.1806777.

28 Laurence A. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393, 1982.

29 Sam Chiu-wai Wong. Tight algorithms for vertex cover with hard capacities on multigraphs and hypergraphs. In *SODA*, pages 2626–2637, 2017.

Faster Algorithms for some Optimization Problems on Collinear Points

Ahmad Biniaz[1]
Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada

Prosenjit Bose[2]
School of Computer Science, Carleton University, Ottawa, Canada

Paz Carmi[3]
Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, Isreal

Anil Maheshwari[4]
School of Computer Science, Carleton University, Ottawa, Canada

Ian Munro[5]
Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada

Michiel Smid[6]
School of Computer Science, Carleton University, Ottawa, Canada

Abstract

We propose faster algorithms for the following three optimization problems on n collinear points, i.e., points in dimension one. The first two problems are known to be NP-hard in higher dimensions.

1. *Maximizing total area of disjoint disks*: In this problem the goal is to maximize the total area of nonoverlapping disks centered at the points. Acharyya, De, and Nandy (2017) presented an $O(n^2)$-time algorithm for this problem. We present an optimal $\Theta(n)$-time algorithm.
2. *Minimizing sum of the radii of client-server coverage*: The n points are partitioned into two sets, namely clients and servers. The goal is to minimize the sum of the radii of disks centered at servers such that every client is in some disk, i.e., in the coverage range of some server. Lev-Tov and Peleg (2005) presented an $O(n^3)$-time algorithm for this problem. We present an $O(n^2)$-time algorithm, thereby improving the running time by a factor of $\Theta(n)$.
3. *Minimizing total area of point-interval coverage*: The n input points belong to an interval I. The goal is to find a set of disks of minimum total area, covering I, such that every disk contains at least one input point. We present an algorithm that solves this problem in $O(n^2)$ time.

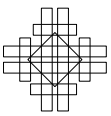
2012 ACM Subject Classification Theory of computation → Computational geometry, Computing methodologies → Optimization algorithms

Keywords and phrases collinear points, range assignment

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.8

Related Version A full version of this paper is available at `https://arxiv.org/abs/1802.09505`

[1] Supported by NSERC and Fields Institute.
[2] Supported by NSERC.
[3] Partially supported by Grant 2016116 from the United States – Israel Binational Science Foundation.
[4] Supported by NSERC.
[5] Supported by NSERC and Canada Research Chairs Program.
[6] Supported by NSERC.



1 Introduction

Range assignment is a well-studied class of geometric optimization problems that arises in wireless network design, and has a rich literature. The task is to assign transmission ranges to a set of given base station antennas such that the resulting network satisfies a given property. The antennas are usually represented by points in the plane. The coverage region of an antenna is usually represented by a disk whose center is the antenna and whose radius is the transmission range assigned to that antenna. In this model, a range assignment problem can be interpreted as the following problem. Given a set of points in the plane, we must choose a radius for each point, so that the disks with these radii satisfy a given property.

Let $P = \{p_1, \ldots, p_n\}$ be a set of n points in the d-dimensional Euclidean space. A *range assignment* for P is an assignment of a transmission range $r_i \geqslant 0$ (radius) to each point $p_i \in P$. The cost of a range assignment, representing the power consumption of the network, is defined as $C = \sum_i r_i^\alpha$ for some constant $\alpha \geqslant 1$. We study the following three range assignment problems on a set of points on a straight-line (1-dimensional Euclidean space).

Problem 1 Given a set of collinear points, maximize the total area of nonoverlapping disks centered at these points. The nonoverlapping constraint requires $r_i + r_{i+1}$ to be no larger than the Euclidean distance between p_i and p_{i+1}, for every $i \in \{1, \ldots, n-1\}$.

Problem 2 Given a set of collinear points that is partitioned into two sets, namely clients and servers, the goal is to minimize the sum of the radii of disks centered at the servers such that every client is in some disk, i.e., every client is covered by at least one server.

Problem 3 Given a set of input points on an interval, minimize the total area of disks covering the entire interval such that every disk contains at least one input point.

In Problem 1 we want to maximize $\sum r_i^2$, in Problem 2 we want to minimize $\sum r_i$, and in Problem 3 we want to minimize $\sum r_i^2$. These three problems are solvable in polynomial time in 1-dimension. Both Problem 1 and Problem 2 are NP-hard in dimension d, for every $d \geqslant 2$, and both have a PTAS [2, 3, 4].

Acharyya et al. [2] showed that Problem 1 can be solved in $O(n^2)$ time. Eppstein [8] proved that an alternate version of this problem, where the goal is to maximize the sum of the radii, can be solved in $O(n^{2-1/d})$ time for any constant dimension d. Bilò et al. [4] showed that Problem 2 is solvable in polynomial time by reducing it to an integer linear program with a totally unimodular constraint matrix. Lev-Tov and Peleg [10] presented an $O(n^3)$-time algorithm for this problem. They also presented a linear-time 4-approximation algorithm. Alt et al. [3] improved the ratio of this linear-time algorithm to 3. They also presented an $O(n \log n)$-time 2-approximation algorithm for Problem 2. Chambers et al. [7] studied a variant of Problem 3—on collinear points—where the disks centered at input points; they showed that the best solution with two disks gives a 5/4-approximation. Carmi et al. [6] studied a similar version of the problem for points in the plane.

1.1 Our contributions

In this paper we study Problems 1-3. In Section 2, we present an algorithm that solves Problem 1 in linear time, provided that the points are given in sorted order along the line. This improves the previous best running time by a factor of $\Theta(n)$. In Section 3, we present an algorithm that solves Problem 2 in $O(n^2)$ time; this also improves the previous best running time by a factor of $\Theta(n)$. In Section 4, first we present a simple $O(n^3)$ algorithm for Problem 3. Then with a more involved proof, we show how to improve the running time to $O(n^2)$.

2 Problem 1: disjoint disks with maximum area

In this section we study Problem 1: Let $P = \{p_1, \ldots, p_n\}$ be a set of $n \geqslant 3$ points on a straight-line ℓ that are given in sorted order. We want to assign to every $p_i \in P$ a radius r_i such that the disks with the given radii do not overlap and their total area, or equivalently $\sum r_i^2$, is as large as possible. Acharyya et al. [1] showed how to obtain such an assignment in $O(n^2)$ time. We show how to obtain such an assignment in linear time.

▶ **Theorem 1.** *Given n collinear points in sorted order in the plane, in $\Theta(n)$ time, we can find a set of nonoverlapping disks centered at these points that maximizes the total area of the disks.*

With a suitable rotation we assume that ℓ is horizontal. Moreover, we assume that $p_1, \ldots, p_n$ is the sequence of points of P in increasing order of their x-coordinates. We refer to a set of nonoverlapping disks centered at points of P as a *feasible solution*. We refer to the disks in a feasible solution S that are centered at $p_1, \ldots, p_n$ as $D_1, \ldots, D_n$, respectively. Also, we denote the radius of D_i by r_i; it might be that $r_i = 0$. For a feasible solution S we define $\alpha(S) = \sum r_i^2$. Since the total area of the disks in S is $\pi \cdot \alpha(S)$, hereafter, we refer to $\alpha(S)$ as the total area of disks in S. We call D_i a *full disk* if it has p_{i-1} or p_{i+1} on its boundary, a *zero disk* if its radius is zero, and a *partial disk* otherwise. For two points p_i and p_j, we denote the Euclidean distance between p_i and p_j by $|p_ip_j|$.

We briefly review the $O(n^2)$-time algorithm of Acharyya et al. [1]. First, compute a set $\mathcal{D}$ of disks centered at points of P, which is the superset of every optimal solution. For every disk $D \in \mathcal{D}$, that is centered at a point $p \in P$, define a weighted interval I whose length is $2r$, where r is the radius of D, and whose center is p. Set the weight of I to be r^2. Let $\mathcal{I}$ be the set of these intervals. The disks corresponding to the intervals in a maximum weight independent set of the intervals in $\mathcal{I}$ forms an optimal solution to Problem 1. By construction, these disks are nonoverlapping, centered at $p_1, \ldots, p_n$, and maximize the total area. Since the maximum weight independent set of m intervals that are given in sorted order of their left endpoints can be computed in $O(m)$ time [9], the time complexity of the above algorithm is essentially dominated by the size of $\mathcal{D}$. Acharyya et al. [1] showed how to compute such a set $\mathcal{D}$ of size $\Theta(n^2)$ and order the corresponding intervals in $O(n^2)$ time. Therefore, the total running time of their algorithm is $O(n^2)$.

We show how to improve the running time to $O(n)$. In fact we show how to find a set $\mathcal{D}$ of size $\Theta(n)$ and order the corresponding intervals in $O(n)$ time, provided that the points of P are given in sorted order.

2.1 Computation of $\mathcal{D}$

In this section we show how to compute a set $\mathcal{D}$ with a linear number of disks such that every disk in an optimal solution for Problem 1 belongs to $\mathcal{D}$.

Our set $\mathcal{D}$ is the union of three sets F, $\overrightarrow{D}$, and $\overleftarrow{D}$ of disks that are computed as follows. The set F contains $2n$ disks representing the full disks and zero disks that are centered at points of P. We compute $\overrightarrow{D}$ by traversing the points of P from left to right as follows; the computation of $\overleftarrow{D}$ is symmetric. For each point p_i with $i \in \{2, \ldots, n-1\}$ we define its *signature* $s(p_i)$ as

$$s(p_i) = \begin{cases} + & \text{if } |p_{i-1}p_i| \leqslant |p_ip_{i+1}| \\ - & \text{if } |p_{i-1}p_i| > |p_ip_{i+1}|. \end{cases}$$

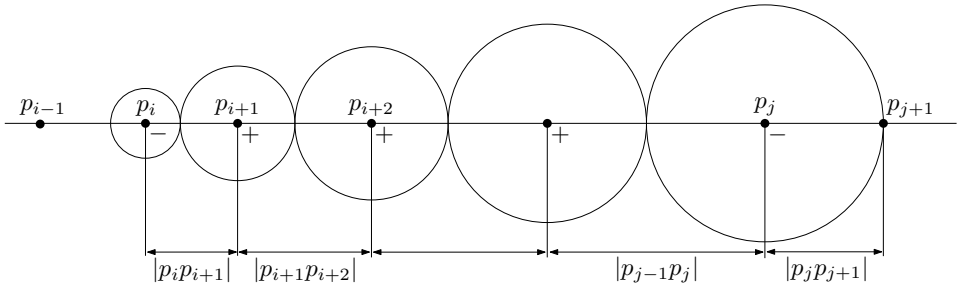


Figure 1 Illustration of a sequence $s(p_i), \ldots, s(p_j) = - + + + -$ in Δ; construction of $\overrightarrow{D}$.

Set $s(p_1) = -$ and $s(p_n) = +$. We refer to the sequence $\mathcal{S} = s(p_1), \ldots, s(p_n)$ as the *signature sequence* of P. Let Δ be the multiset that contains all contiguous subsequences $s(p_i), \ldots, s(p_j)$ of $\mathcal{S}$, with $i < j$, such that $s(p_i) = s(p_j) = -$, and $s(p_k) = +$ for all $i < k < j$; if $j = i + 1$, then there is no k. For example, if $\mathcal{S} = - + + - + + + - - - + - - + +$, then $\Delta = \{- + +-, - + + + -, --, --, - + -, --\}$. Observe that for every sequence $s(p_i), \ldots, s(p_j)$ in Δ we have that

$$|p_ip_{i+1}| \leqslant |p_{i+1}p_{i+2}| \leqslant |p_{i+2}p_{i+3}| \leqslant \cdots \leqslant |p_{j-1}p_j|, \quad \text{and} \quad |p_{j-1}p_j| > |p_jp_{j+1}|.$$

Every plus sign in $\mathcal{S}$ belongs to at most one sequence in Δ, and every minus sign in $\mathcal{S}$ belongs to at most two sequences in Δ. Therefore, the size of Δ (the total length of its sequences) is at most $2n$. For each sequence $s(p_i), \ldots, s(p_j)$ in Δ we add some disks to $\overrightarrow{D}$ as follows. Consider the full disk D_j at p_j. Iterate on $k = j - 1, j - 2, \ldots, i$. In each iteration, consider the disk D_k that is centered at p_k and touches D_{k+1}. If D_k does not contain p_{k-1} and its area is smaller than the area of D_{k+1}, then add D_k to $\overrightarrow{D}$ and proceed to the next iteration, otherwise, terminate the iteration. See Figure 1. This finishes the computation of $\overrightarrow{D}$. Notice that $\overrightarrow{D}$ contains at most $n - 1$ disks. The computation of $\overleftarrow{D}$ is symmetric; it is done in a similar way by traversing the points from right to left (all the $+$ signatures become $-$ and vice versa).

The number of disks in $\mathcal{D} = F \cup \overrightarrow{D} \cup \overleftarrow{D}$ is at most $4n - 2$. The signature sequence $\mathcal{S}$ can be computed in linear time. Having $\mathcal{S}$, we can compute the multiset Δ, the disks in $\overrightarrow{D}$ as well as the corresponding intervals, as in [1] and described before, in sorted order of their left endpoints in total $O(n)$ time. Then the sorted intervals corresponding to circles in $\mathcal{D}$ can be computed in linear-time by merging the sorted intervals that correspond to sets F, $\overrightarrow{D}$, and $\overleftarrow{D}$. It remains to show that $\mathcal{D}$ contains an optimal solution for Problem 1. To that end, we first prove two lemmas about the structural properties of an optimal solution.

▶ **Lemma 2.** *Every feasible solution S for Problem 1 can be converted to a feasible solution S' where D_1 and D_n are full disks and $\alpha(S') \geqslant \alpha(S)$.*

Proof. Recall that $n \geqslant 3$. We prove this lemma for D_1; the proof for D_n is similar. Since S is a feasible solution, we have that $r_1 + r_2 \leqslant |p_1p_2|$. Let S' be a solution that is obtained from S by making D_1 a full disk and D_2 a zero disk. Since we do not increase the radius of D_2, it does not overlap D_3, and thus, S' is a feasible solution. In S', the radius of D_1 is $|p_1p_2|$, and we have that $r_1^2 + r_2^2 \leqslant (r_1 + r_2)^2 \leqslant |p_1p_2|^2$. This implies that $\alpha(S') \geqslant \alpha(S)$. ◀

▶ **Lemma 3.** *If D_i, with $1 < i < n$, is a partial disk in an optimal solution, then $r_i < \max(r_{i-1}, r_{i+1})$.*

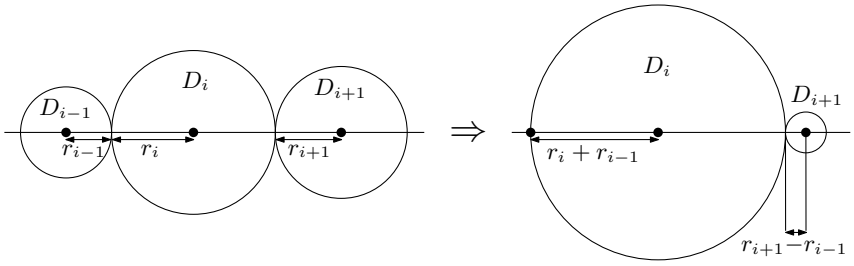


Figure 2 Illustration of the proof of Lemma 3.

Proof. The proof is by contradiction; let S be such an optimal solution for which $r_i \geqslant \max(r_{i-1}, r_{i+1})$. First assume that D_i touches at most one of D_{i-1} and D_{i+1}. By slightly enlarging D_i and shrinking its touching neighbor we can increase the total area of S. Without loss of generality suppose that D_i touches D_{i-1}. Since $r_i \geqslant r_{i-1}$,

$$(r_i + \epsilon)^2 + (r_{i-1} - \epsilon)^2 = r_i^2 + r_{i-1}^2 + 2(r_i\epsilon - r_{i-1}\epsilon + \epsilon^2) > r_i^2 + r_{i-1}^2 > 0,$$

for any $\epsilon > 0$. This contradicts optimality of S. Now, assume that D_i touches both D_{i-1} and D_{i+1}, and that $r_{i-1} \leqslant r_{i+1}$. See Figure 2. We obtain a solution S' from S by enlarging D_i as much as possible, and simultaneously shrinking both D_{i-1} and D_{i+1}. This makes D_{i-1} a zero disk, D_i a full disk, D_{i+1} a zero or a partial disk, and does not change the other disks. The difference between the total areas of S' and S is

$$\big((r_i + r_{i-1})^2 + (r_{i+1} - r_{i-1})^2\big) - (r_{i-1}^2 + r_i^2 + r_{i+1}^2) = r_{i-1}^2 + 2r_{i-1}(r_i - r_{i+1}) > 0;$$

this inequality is valid since $r_i \geqslant r_{i+1} \geqslant r_{i-1} > 0$. This contradicts the optimality of S. ◀

▶ **Lemma 4.** *The set $\mathcal{D}$ contains an optimal solution for Problem 1.*

Proof. It suffices to show that every disk D_k, which is centered at p_k, in an optimal solution $S = \{D_1, \ldots, D_n\}$ belongs to $\mathcal{D}$. By Lemma 2, we may assume that both D_1 and D_n are full disks. If D_k is a full disk or a zero disk, then it belongs to F. Assume that D_k is a partial disk. Since S is optimal, D_k touches at least one of D_{k-1} and D_{k+1}, because otherwise we could enlarge D_k.

First assume that D_k touches exactly one disk, say D_{k+1}. We are going to show that D_k belongs to $\overrightarrow{D}$ (If D_k touches only D_{k-1}, by a similar reasoning we can show that D_k belongs to $\overleftarrow{D}$). Notice that $r_k < r_{k+1}$, because otherwise we could enlarge D_k and shrink D_{k+1} simultaneously to increase $\alpha(S)$, which contradicts the optimality of S. Since D_k is partial and touches D_{k+1}, we have that D_{k+1} is either full or partial. If D_{k+1} is full, then it has p_{k+2} on its boundary, and thus $s(p_{k+1}) = -$. By our definition of Δ, for some $i < k+1$, the sequence $s(p_i), \ldots, s(p_{k+1})$ belongs to Δ. Then by our construction of $\overrightarrow{D}$ both D_{k+1} and D_k belong to $\overrightarrow{D}$, where $k+1$ plays the role of j. Assume that D_{k+1} is partial. Then D_{k+2} touches D_{k+1}, because otherwise we could enlarge D_{k+1} and shrink D_k simultaneously to increase $\alpha(S)$. Recall that $r_k < r_{k+1}$. Lemma 3 implies that $r_{k+1} < r_{k+2}$. This implies that $|p_k p_{k+1}| < |p_{k+1} p_{k+}|$, and thus $s(p_{k+1}) = +$. Since D_{k+1} is partial and touches D_{k+2}, we have that D_{k+2} is either full or partial. If D_{k+2} is full, then it has p_{k+3} on its boundary, and thus $s(p_{k+2}) = -$. By a similar reasoning as for D_{k+1} based on the definition of Δ and $\overrightarrow{D}$, we get that D_{k+2}, D_{k+1}, and D_k are in $\overrightarrow{D}$. If D_{k+2} is partial, then it touches D_{k+3} and again by Lemma 3 we have $r_{k+2} < r_{k+3}$ and consequently $s(p_{k+2}) = +$. By repeating this

process, we stop at some point p_j, with $j \leqslant n-2$, for which D_j is a full disk, $r_{j-1} < r_j$, and $s(p_j) = -$; notice that such a j exists because D_n is a full disk and consequently D_{n-1} is a zero disk. To this end we have that $s(p_k) \in \{+,-\}$, $s(p_j) = -$, and $s(p_{k+1}), \ldots, s(p_{j-1})$ is a plus sequence. Thus, $s(p_k), \ldots, s(p_j)$ is a subsequence of some sequence $s(p_i), \ldots, s(p_j)$ in Δ. Our construction of $\overrightarrow{D}$ implies that all disks $D_k, \ldots, D_j$ belong to $\overrightarrow{D}$.

Now assume that D_k touches both D_{k-1} and D_{k+1}. By Lemma 3 we have that D_k is strictly smaller than the largest of these disks, say D_{k+1}. By a similar reasoning as in the previous case we get that $D_k \in \overrightarrow{D}$. ◀

3 Problem 2: client-server coverage with minimum radii

In this section we study Problem 2: Let $P = \{p_1, \ldots, p_n\}$ be a set of n points on a straight-line ℓ that is partitioned into two sets, namely clients and servers. We want to assign to every server in P a radius such that the disks with these radii cover all clients and the sum of their radii is as small as possible. Bilò et al. [4] showed that this problem can be solved in polynomial time. Lev-Tov and Peleg [10] showed how to obtain such an assignment in $O(n^3)$ time. Alt et al. [3] presented an O(n log n)-time 2-approximation algorithm for this problem. We show how to solve this problem optimally in $O(n^2)$ time.

▶ **Theorem 5.** *Given a total of n collinear clients and servers, in $O(n^2)$ time, we can find a set of disks centered at servers that cover all clients and where the sum of the radii of the disks is minimum.*

Without loss of generality assume that ℓ is horizontal, and that $p_1, \ldots, p_n$ is the sequence of points of P in increasing order of their x-coordinates. We refer to a disk with radius zero as a *zero disk*, to a set of disks centered at servers and covering all clients as a *feasible solution*, and to the sum of the radii of the disks in a feasible solution as its *cost*. We denote the radius of a disk D by $r(D)$, and denote by $D(p,q)$ a disk that is centered at the point p with the point q on its boundary.

We describe a top-down dynamic programming algorithm that maintains a table T with n entries $T(1), \ldots, T(n)$. Each table entry $T(k)$ represents the cost of an optimal solution for the subproblem that consists of points $p_1, \ldots, p_k$. The optimal cost of the original problem will be stored in $T(n)$; the optimal solution itself can be recovered from T. In the rest of this section we show how to solve a subproblem $p_1, \ldots, p_k$. In fact, we show how to compute $T(k)$ recursively by a top-down dynamic programming algorithm. To that end, we first describe our three base cases:

- There is no client. In this case $T(k) = 0$.
- There are some clients but no server. In this case $T(k) = +\infty$.
- There are some clients and exactly one server, say s. In this case $T(k)$ is the radius of the smallest disk that is centered at s and covers all the clients.

Assume that the subproblem $p_1, \ldots, p_k$ has at least one client and at least two servers. We are going to derive a recursion for $T(k)$.

▶ **Observation 6.** *Every disk in any optimal solution has a client on its boundary.*

▶ **Lemma 7.** *No disk contains the center of some other non-zero disk in an optimal solution.*

Proof. Our proof is by contradiction. Let D_i and D_j be two disks in an optimal solution such that D_i contains the center of D_j. Let p_i and p_j be the centers of D_i and D_j, respectively, and r_i and r_j be the radii of D_i and D_j, respectively. See Figure 3(a). Since D_i contains

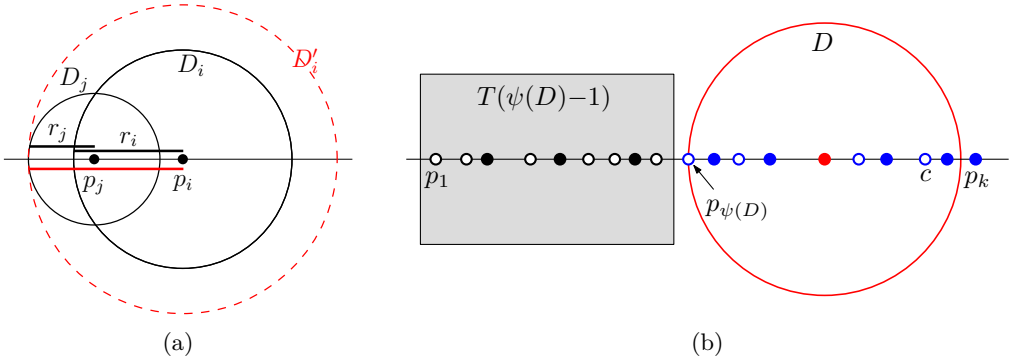


Figure 3 (a) Illustration of the proof of Lemma 7. (b) Clients are shown by small circles, and servers are shown by small disks. $p_{\psi(D)}$ is the leftmost point (client or server) in D.

p_j, we have $r_i > |p_ip_j|$. Let D'_i be the disk of radius $|p_ip_j| + r_j$ that is centered at p_i. Notice that D'_i covers all the clients that are covered by $D_i \cup D_j$. By replacing D_i and D_j with D'_i we obtain a feasible solution whose cost is smaller than the optimal cost, because $|p_ip_j| + r_j < r_i + r_j$. This contradicts the optimality of the initial solution. ◀

Let c be the rightmost client in $p_1, \ldots, p_k$. For a disk D that covers c, let $\psi(D) \in \{1, \ldots, k\}$ be the smallest index for which the point $p_{\psi(D)}$ is in the interior or on the boundary of D, i.e., $\psi(D)$ is the index of the leftmost point of $p_1, \ldots, p_k$ that is in D. See Figure 3(b).

We claim that only one disk in an optimal solution can cover c, because, if two disks cover c then if their centers lie on the same side of c, we get a contradiction to Lemma 7, and if their centers lie on different sides of c, then by removing the disk whose center is to the right of c we obtain a feasible solution with smaller cost. Let S^* be an optimal solution (with minimum sum of the radii) that has a maximum number of non-zero disks. Let D^* be the disk in S^* that covers c. All other clients in $p_{\psi(D^*)}, \ldots, p_k$ are also covered by D^*, and thus, they do not need to be covered by any other disk. As a consequence of Lemma 7, the servers that are in D^* and the servers that lie to the right of D^* cannot be used to cover any clients in $p_1, \ldots, p_{\psi(D^*)-1}$. Therefore, if we have D^*, then the problem reduces to a smaller instance that consists of the points to the left of D^*, i.e., $p_1, \ldots, p_{\psi(D^*)-1}$. See Figure 3(b). Thus, the cost of the optimal solution for the subproblem $p_1, \ldots, p_k$ can be computed as $T(k) = T(\psi(D^*) - 1) + r(D^*)$.

In the rest of this section we compute a set $\mathcal{D}_k$ of $O(k)$ disks each of them covering c. Then we claim that D^* belongs to $\mathcal{D}_k$. Therefore, we can compute $T(k)$ by this recursion:

$$T(k) = \min\{T(\psi(D) - 1) + r(D) : D \in \mathcal{D}_k\}.$$

We compute $\mathcal{D}_k$ in two phases. In the first phase, for every server s we add the disk $D(s, c)$ to $\mathcal{D}_k$. In the second phase, we consider the servers that are to the left of c and the servers that are to the right of c separately. Let $s_1, s_2, \ldots, s_m$ be the sequence of all servers that are to the left of c; see Figure 4. For every $i \in \{1, \ldots, m\}$, let s'_i be the left intersection point of the boundary of $D(s_i, c)$ with ℓ. Set $s'_0 = -\infty$. Let C_i be the longest sequence of consecutive clients between s'_{i-1} and s'_i that lie just before s'_i; there is no server between s'_i and the leftmost point of C_i. For every client $c' \in C_i$ we add the disk $D(s_i, c')$ to $\mathcal{D}_k$ as in Figure 4. Now we consider the servers $s_{m+1}, s_{m+2}, \ldots$, which are to the right of c. See Figure 4. Notice that the optimal solution does not contain a disk D that is centered at any of the servers $s_{m+2}, s_{m+3}, \ldots$ because otherwise we could replace D by a smaller disk

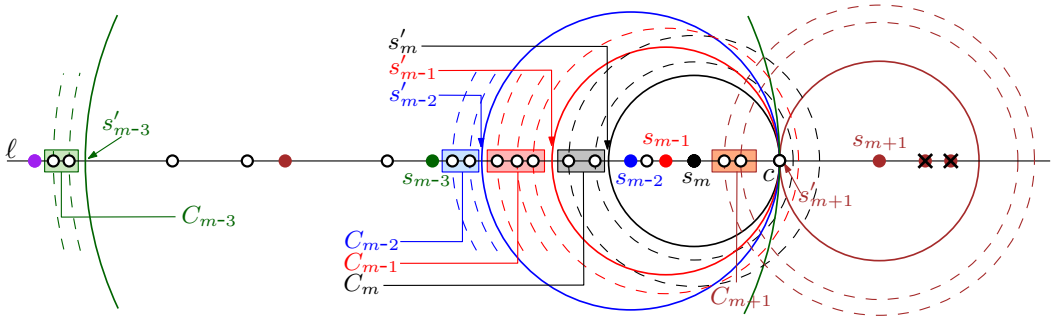


Figure 4 Computation of $\mathcal{D}_k$; small circles are clients and small disks are servers.

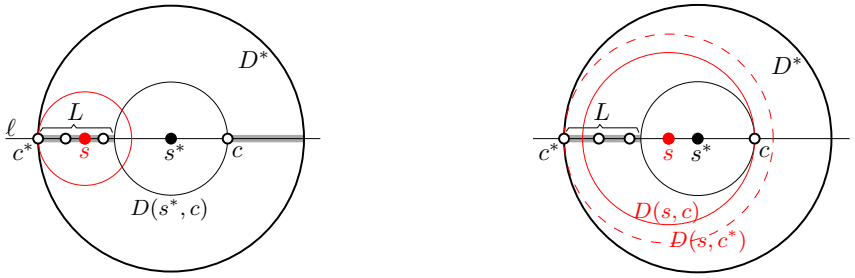


Figure 5 Illustration of the proof of Lemma 8: (a) The server s lies on L. (b) The boundary of $D(s, c)$ intersects L.

D' centered at s_{m+1} such that D' covers the same clients that are covered by D'. Thus, we simply discard $s_{m+2}, s_{m+3}, \ldots$. For s_{m+1}, we define C_{m+1} in a similar way that we defined C_i (notice that here we have $s'_{m+1} = c$), and then for each client $c' \in C_{m+1}$ we add $D(s_{m+1}, c')$ to $\mathcal{D}_k$; see Figure 4. This finishes the computation of $\mathcal{D}_k$. In the first phase we added one disk to $\mathcal{D}_k$ for every server. In the second phase we added one disk for every client in C_i for all $i \in \{1, \ldots, m+1\}$. The sets C_i are pairwise disjoint because each C_i contains some clients that lie between s'_{i-1} and s_i. Thus the total number of disks added to $\mathcal{D}_k$ in the second phase is at most the number of clients. Therefore the total number of disks in $\mathcal{D}_k$ is at most k. The set $\mathcal{D}_k$, and consequently the entry $T(k)$ can be computed in $O(k)$ time. Therefore, our dynamic programming algorithm computes all entries of T in $O(n^2)$ time.

To complete the correctness proof of our algorithm it only remains to show that D^* belongs to $\mathcal{D}_k$. If D^* has c on its boundary, then D^* has been added to $\mathcal{D}_k$ in the first phase of the computation of $\mathcal{D}_k$. Assume that D^* does not have c on its boundary. Let c^*, with $c^* \neq c$, be the client on the boundary of D^* (such a client exists by Observation 6). Let s^* be the center of D^*. Since c is the rightmost client and c^* is on the boundary of D^*, we have that c^* is to the left of s^* (but c can be to the left or to the right of s^*). See Figure 5. Observe that $D(s^*, c)$ is in the interior of D^*. The intersection of ℓ with the disk difference $D^* \setminus D(s^*, c)$ consists of two line segments; let L be the one to the left.

▶ **Lemma 8.** *There is no server on L and there is no server s such that the boundary of $D(s, c)$ intersects L.*

Proof. We prove both statements by contradiction. To prove the first statement assume that L contains a server s; see Figure 5(a). We can replace D^* by $D(s^*, c)$ and the smallest disk that is centered at s and covers all the clients on L. These two new disks cover all the

clients that have been covered by D^*. If s is strictly inside L, then sum of the radii of these two disks is smaller than the radius of D^*, thereby this replacement reduces the optimal cost, which contradicts the optimality of S^*. If s is on the right endpoint of L, then the sum of the radii of these two disks is equal to the radius of D^*, thereby this replacement increases the number of non-zero disks without increasing the optimal cost; this contradicts our choice of S^* (notice that, by Lemma 7, s has a zero disk in S^*).

To prove the second statement let s be a server such that $D(s, c)$ intersects L; see Figure 5(b). Since $D(s, c)$ intersects L, s lies to the left of s^*. In this configuration, $D(s, c)$ is contained in $D(s, c^*)$ (no matter if c is to the left or to the right of s). Also, $D(s, c^*)$ is smaller than D^*, and covers all the clients that are covered by D^*. Thus, by replacing D^* with $D(s, c^*)$ we obtain a feasible solution whose cost is smaller than the optimal cost, which is a contradiction. ◀

Let C_L be the set of all clients on L (including c^*). By the first statement of Lemma 8 the clients in C_L are consecutive. Let $s^* = s_j$ for some $j \in \{1, \ldots, m+1\}$. Then by our definition of L, the clients in C_L lie just before s'_j. By the second statement of Lemma 8 the clients in C_L are to the right of s'_{j-1}. These constraints imply that $C_L \subseteq C_j$. Therefore, the disk $D(s_j, c^*) = D^*$ is contained in $\mathcal{D}_k$, since it was added in the second phase of the construction. This finishes the correctness proof.

4 Problem 3: point-interval coverage with minimum area

Let $I = [a, b]$ be an interval on the x-axis in the plane. We say that a set of disks *covers* I if I is a subset of the union of the disks in this set. Let P be a set of n points on I such that a and b are in P. A *point-interval coverage* for the pair (P, I) is a set S of disks that cover I such that (i) the disks in S have their centers on I and (ii) every disk in S contains at least one point of P. See Figure 6. The *point-interval coverage* problem is to find such a set of disks with minimum total area. In this section we show how to solve this problem in $O(n^2)$ time.

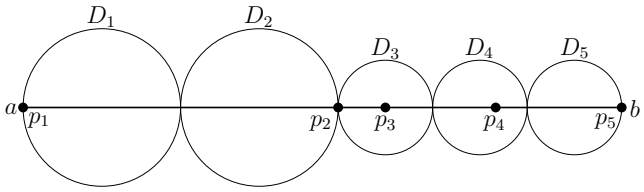


Figure 6 The minimum point-interval coverage for $(\{p_1, \ldots, p_5\}, [a, b])$; each p_i is assigned to D_i.

▶ **Theorem 9.** *Given n points on an interval, in $O(n^2)$ time, we can find a set of disks covering the entire interval such that every disk contains at least one point and where the total area of the disks is minimum.*

If we drop condition (ii), then the problem can be solved in linear time by using Observation 14 (which is stated below). Let Problem 3′ be a version of the point-interval coverage problem with an additional constraint that (iii) every point $p \in P$ is assigned to exactly one of the disks in S that contains p. Notice that any solution for Problem 3′ is also a solution for Problem 3. Conversely, any solution for Problem 3 can be transformed to a solution for Problem 3′ by assigning every point to one of the disks containing it. Thus, these two problems are equivalent. Therefore, without loss of generality, in the rest of this section we

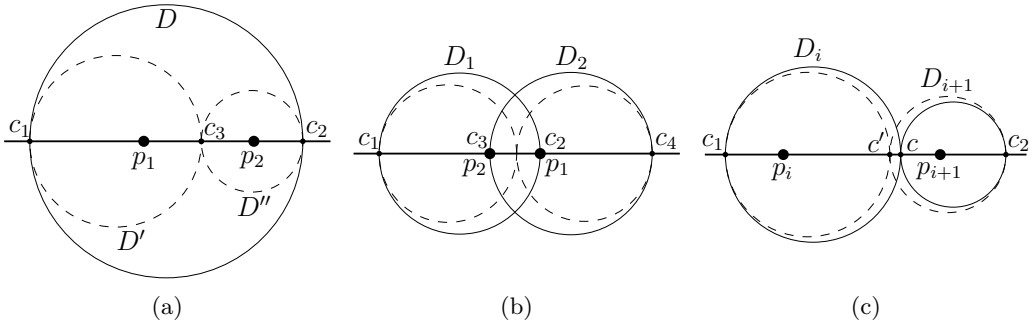


Figure 7 Illustrations of the proofs of (a) Lemma 10, (b) Lemma 11, and (c) Lemma 12.

study the version of the point-interval coverage with three constraints (i), (ii), and (iii). First we prove some lemmas about the structural properties of an optimal point-interval coverage. We say that a disk is *anchored* at a point p if it has p on its boundary. We say that two intersecting disks *touch* each other if their intersection is exactly one point, and we say that they *overlap* otherwise.

▶ **Lemma 10.** *In any optimal solution for the point-interval coverage problem, exactly one point is assigned to each disk.*

Proof. Our proof is by contradiction. Consider a disk D in an optimal solution that is assigned two points p_1 and p_2. Without loss of generality assume that p_1 is to the left of p_2. Let c_1 and c_2 be the two intersection points of the boundary of D with the x-axis, and let c_3 be a point on the x-axis that is between p_1 and p_2. Let D' and D'' be the disks with diameters c_1c_3 and c_2c_3, respectively; see Figure 7(a). The total area of D' and D'' is smaller than the total area of D. Also $D' \cup D''$ covers the same interval as D does. Remove D from the optimal set and add D' and D'' to the resulting set. Assign p_1 to D' and p_2 to D''. This gives a solution with smaller total area, which is a contradiction. ◀

▶ **Lemma 11.** *There is no pair of overlapping disks in any optimal solution for the point-interval coverage problem.*

Proof. Our proof is by contradiction. Consider two overlapping disks D_1 and D_2 in an optimal solution. Let p_1 and p_2 denote the points that are assigned to D_1 and D_2, respectively. We differentiate between the following two cases.

- *D_1 is a subset of D_2, or vice versa.* Assume that D_1 is a subset of D_2. Let c_1 and c_2 be the two intersection points of the boundary of D_2 with the x-axis. Let D' and D'' be the disks with diameters p_1c_1 and p_1c_2. The total area of D' and D'' is smaller than the total area of D_1 and D_2. Moreover, $D' \cup D''$ covers the same interval as $D_1 \cup D_2$ does. Remove D_1 and D_2 from the optimal set and add D' and D'' to the resulting set. Assign p_2 to one of D' and D'' that contains p_2, and assign p_1 to the other disk. This results a solution whose total area is smaller than the optimal area, which is a contradiction.
- *D_1 is not a subset of D_2, nor vice versa.* See Figure 7(b). Let c_1 and c_2 be the left and right intersection points of the boundary of D_1 with the x-axis, respectively. Let c_3 and c_4 be the left and right intersection points of the boundary of D_2 with x-axis, respectively. Assume that c_1 is to the left of c_4; this implies c_1, c_3, c_2, c_4 is the sorted sequence of these points from left to right. If $p_1 \neq c_2$, then we shrink D_1 (while anchored at c_1) by a small amount and reduce the total area of the optimal set, which is a contradiction. Assume

that $p_1 = c_2$; similarly, assume that $p_2 = c_3$. In this configuration we shrink D_1 (while anchored at c_1) and D_2 (while anchored at c_4) simultaneously until they touch each other as in Figure 7(b). Then we assign p_2 to D_1, and p_1 to D_2. This gives a valid solution whose total area is smaller than the optimal area, which is a contradiction. ◀

Lemma 10 implies that the number of disks in every optimal solution—for the interval coverage problem—is equal to the number of points in P, and Lemma 11 implies that these disks can touch each other but do not overlap. This enables us to order the disks of every optimal solution from left to right such that any two consecutive disks touch each other; let $D_1, \ldots, D_n$ be this ordering. Let $p_1, \ldots, p_n$ be the points of P from left to right. Then for every $i \in \{1, \ldots, n\}$, the point p_i is assigned to the disk D_i; see Figure 6.

▶ **Lemma 12.** *In any optimal solution, if the intersection point of D_i and D_{i+1} does not belong to P, then D_i and D_{i+1} have equal radius.*

Proof. Let c be the intersection point of D_i and D_{i+1}. Let c_1 be the left intersection point of the boundary of D_i with the x-axis, and let c_2 be the right intersection point of the boundary of D_{i+1} with the x-axis; see Figure 7(c). We proceed by contradiction, and assume, without loss of generality, that D_{i+1} is smaller than D_i. We shrink D_i (while anchored at c_1) and enlarge D_{i+1} (while anchored at c_2) simultaneously by a small value. This gives a valid solution whose total area is smaller than the optimal area, because our gain in the area of D_{i+1} is smaller than our loss from the area of D_i. This contradicts the optimality of our initial solution. ◀

The following lemma and observation play important roles in our algorithm for the point-interval coverage problem, which we describe later.

▶ **Lemma 13.** *Let $R > 0$ be a real number, and $r_1, r_2, \ldots, r_k$ be a sequence of positive real numbers such that $\sum_{i=1}^{k} r_i = R$. Then*

$$\sum_{i=1}^{k} r_i^2 \geq \sum_{i=1}^{k} (R/k)^2 = R^2/k, \tag{1}$$

i.e., the sum on the left-hand side of (1) is minimum if all r_i are equal to R/k.

Proof. If f is a convex function, then—by Jensen's inequality—we have

$$f\left(\sum_{i=1}^{k} \frac{r_i}{k}\right) \leqslant \sum_{i=1}^{k} \frac{f(r_i)}{k}.$$

Since the function $f(x) = x^2$ is convex, it follows that

$$\left(\frac{R}{k}\right)^2 = f\left(\frac{R}{k}\right) = f\left(\sum_{i=1}^{k} \frac{r_i}{k}\right) \leqslant \sum_{i=1}^{k} \frac{r_i^2}{k},$$

which, in turn, implies Inequality (1). ◀

The minimum sum of the radii of a set of disks that cover $I = [a, b]$ is $|ab|/2$. The following observation is implied by Lemma 13, by setting $R = |ab|/2$ and $k = n$.

▶ **Observation 14.** *The minimum total area of n disks covering I is obtained by a sequence of n disks of equal radius such that every two consecutive disks touch each other; see Figure 8.*

We refer to the covering of I that is introduced in Observation 14 as the *unit-disk covering* of I with n disks. Such a covering is called *valid* if it is a point-interval coverage for (P, I).

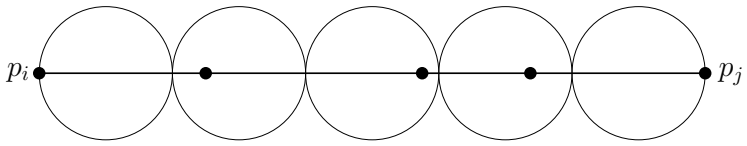


Figure 8 A valid unit-disk covering.

4.1 A dynamic-programming algorithm

In this subsection we present an $O(n^3)$-time dynamic-programming algorithm for the point-interval coverage problem. In the full version of the paper [5] we present a more involved dynamic-programming algorithm that improves the running time to $O(n^2)$.

First, we review some properties of an optimal solution for the point-interval coverage problem that enable us to present a top-down dynamic programming algorithm. Let $C^* = D_1, \ldots, D_n$ be the sequence of n disks in an optimal solution for this problem. Recall that as a consequence of Lemma 11, the intersection of every two consecutive disks in C^* is a point. If there is no $k \in \{1, \ldots, n-1\}$ for which the intersection point of D_k and D_{k+1} belongs to P, then Lemma 12 implies that all disks in C^* have equal radius, and thus, C^* is a valid unit-disk covering. Assume that for some $k \in \{1, \ldots, n-1\}$ the intersection point of D_k and D_{k+1} is a point $p \in P$. Notice that p is assigned to either D_k or D_{k+1}; this implies either $p = p_k$ or $p = p_{k+1}$. In either case, C^* is the union of the optimal solutions for two smaller problem-instances (P_1, I_1) and (P_2, I_2) where $I_1 = [a, p]$, $I_2 = [p, b]$, $P_1 = \{p_1, \ldots, p_k\}$ and $P_2 = \{p_{k+1}, \ldots, p_n\}$.

We define a subproblem (P_{ij}, I_{ij}) and represent it by four indices (i, j, i', j') where $1 \leqslant i < j \leqslant n$ and $i', j' \in \{0, 1\}$. The indices i and j indicate that $I_{ij} = [p_i, p_j]$. The set P_{ij} contains the points of P that are on I_{ij} provided that p_i belongs to P_{ij} if and only if $i' = 1$ and p_j belongs to P_{ij} if and only if $j' = 1$. For example, if $i' = 1$ and $j' = 0$, then $P_{ij} = \{p_i, p_{i+1}, \ldots, p_{j-1}\}$. We define $T(i, j, i', j')$ to be the cost (total area) of an optimal solution for subproblem (i, j, i', j'). The optimal cost of the original problem will be stored in $T(1, n, 1, 1)$. We compute $T(i, j, i', j')$ as follows. If the unit-disk covering is a valid solution for (i, j, i', j'), then by Observation 14 it is optimal, and thus we assign its total area to $T(i, j, i', j')$. Otherwise, as we discussed earlier, there is a point p_k of P with $k \in \{i+1, \ldots, j-1\}$ that is the intersection point of two consecutive disks in the optimal solution. This splits the problem into two smaller subproblems, one to the left of p_k and one to the right of p_k. The point p_k is assigned either to the left subproblem or to the right subproblem. See Figure 9 for an instance in which the unit-disk covering is not valid, and p_k is assigned to the right subproblem. In the optimal solution, p_k is assigned to the one that

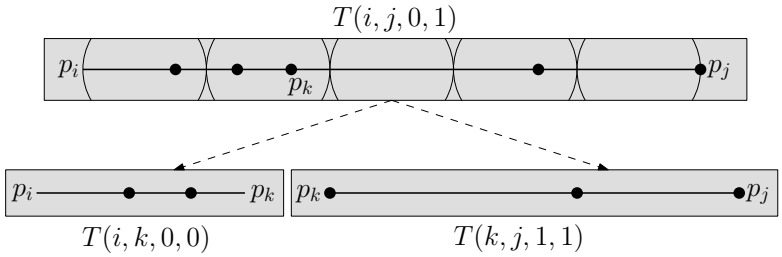


Figure 9 An instance for which the unit-disk covering is not valid.

minimizes the total area, which is

$$T(i,j,i',j') = \min\{T(i,k,i',1) + T(k,j,0,j'), T(i,k,i',0) + T(k,j,1,j')\}.$$

Since we do not know the value of k, we try all possible values and pick the one that minimizes $T(i,j,i',j')$.

There are three base cases for the above recursion. (1) No point of P is assigned to the current subproblem: we assign $+\infty$ to $T(\cdot)$, which implies this solution is not valid. (2) Exactly one point of P is assigned to the current subproblem: we cover $[p_i, p_j]$ with one disk of diameter $|p_i p_j|$ and assign its area to $T(\cdot)$. (3) More than one point of P is assigned to the current subproblem and the unit-disk covering is valid: we assign the total area of this unit-disk covering to $T(\cdot)$.

The total number of subproblems is at most $2 \cdot 2 \cdot \binom{n}{2} = O(n^2)$, because i and j take $\binom{n}{2}$ different values, and each of i' and j' takes two different values. The time to solve each subproblem (i,j,i',j') is proportional to the time for checking the validity of the unit-disk covering for this subproblem plus the iteration of k from $i+1$ to $j-1$; these can be done in total time $O(j-i)$. Thus, the running time of our dynamic programming algorithm is $O(n^3)$.

In the full version of the paper [5] we present a more involved dynamic-programming algorithm that improves the running time to $O(n^2)$. Essentially, our algorithm verifies the validity of the unit-disk coverings for all subproblems $p_i, \ldots, p_j$ in $O(n^2)$ time.

5 Conclusion: an open problem

We considered three optimization problems on collinear points in the plane. Here we present a related open problem: given a set of collinear points, we want to assign to each point a disk, centered at that point, such that the underlying disk graph is connected and the sum of the areas of the disks is minimized. The disk graph has input points as its vertices, and has an edge between two points if their assigned disks intersect. It is not known whether or not this problem is NP-hard. In any dimension $d \geqslant 2$ this problem is NP-hard if an upper bound on the radii of disks is given to us [7].

References

1. Ankush Acharyya, Minati De, and Subhas C. Nandy. Range assignment of base-stations maximizing coverage area without interference. In *Proceedings of the 29th Canadian Conference on Computational Geometry* (*CCCG*), pages 126–131, 2017.
2. Ankush Acharyya, Minati De, Subhas C. Nandy, and Bodhayan Roy. Range assignment of base-stations maximizing coverage area without interference. *CoRR*, abs/1705.09346, 2017.
3. Helmut Alt, Esther M. Arkin, Hervé Brönnimann, Jeff Erickson, Sándor P. Fekete, Christian Knauer, Jonathan Lenchner, Joseph S. B. Mitchell, and Kim Whittlesey. Minimum-cost coverage of point sets by disks. In *Proceedings of the 22nd ACM Symposium on Computational Geometry,* (*SoCG*), pages 449–458, 2006.
4. Vittorio Bilò, Ioannis Caragiannis, Christos Kaklamanis, and Panagiotis Kanellopoulos. Geometric clustering to minimize the sum of cluster sizes. In *Proceedings of the 13th European Symposium on Algorithms,* (*ESA*), pages 460–471, 2005.
5. Ahmad Biniaz, Prosenjit Bose, Paz Carmi, Anil Maheshwari, Ian Munro, and Michiel Smid. Faster algorithms for some optimization problems on collinear points. *CoRR*, abs/1802.09505, 2018.
6. Paz Carmi, Matthew J. Katz, and Joseph S. B. Mitchell. The minimum-area spanning tree problem. *Computational Geometry: Theory and Applications*, 35(3):218–225, 2006.

7 Erin W. Chambers, Sándor P. Fekete, Hella-Franziska Hoffmann, Dimitri Marinakis, Joseph S. B. Mitchell, Srinivasan Venkatesh, Ulrike Stege, and Sue Whitesides. Connecting a set of circles with minimum sum of radii. *Computational Geometry: Theory and Applications*, 68:62–76, 2018.

8 David Eppstein. Maximizing the sum of radii of disjoint balls or disks. In *Proceedings of the 28th Canadian Conference on Computational Geometry* (*CCCG*), pages 260–265, 2016.

9 Ju Yuan Hsiao, Chuan Yi Tang, and Ruay Shiung Chang. An efficient algorithm for finding a maximum weight 2-independent set on interval graphs. *Information Processing Letters*, 43(5):229–235, 1992.

10 Nissan Lev-Tov and David Peleg. Polynomial time approximation schemes for base station coverage with minimum total radii. *Computer Networks*, 47(4):489–501, 2005.

Local Criteria for Triangulation of Manifolds

Jean-Daniel Boissonnat
Université Côte d'Azur, Inria
Sophia Antipolis, France
Jean-Daniel.Boissonnat@inria.fr

Ramsay Dyer
Université Côte d'Azur, Inria
Sophia Antipolis, France
Ramsay.Dyer@inria.fr
https://orcid.org/0000-0001-5083-7181

Arijit Ghosh[1]
Indian Statistical Institute
Kolkata, India
arijitiitkgpster@gmail.com

Mathijs Wintraecken
Université Côte d'Azur, Inria
Sophia Antipolis, France
Mathijs.Wintraecken@inria.fr

Abstract

We present criteria for establishing a triangulation of a manifold. Given a manifold M, a simplicial complex $\mathcal{A}$, and a map H from the underlying space of $\mathcal{A}$ to M, our criteria are presented in local coordinate charts for M, and ensure that H is a homeomorphism. These criteria do not require a differentiable structure, or even an explicit metric on M. No Delaunay property of $\mathcal{A}$ is assumed. The result provides a triangulation guarantee for algorithms that construct a simplicial complex by working in local coordinate patches. Because the criteria are easily verified in such a setting, they are expected to be of general use.

2012 ACM Subject Classification Mathematics of computing → Geometric topology, Mathematics of computing → Mesh generation

Keywords and phrases manifold, simplicial complex, homeomorphism, triangulation

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.9

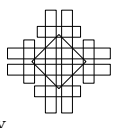
Related Version A full version of the paper is available at `http://arxiv.org/abs/1803.07642`

Funding This work has been partially funded by the European Research Council under the European Union's ERC Grant Agreement number 339025 GUDHI (Algorithmic Foundations of Geometric Understanding in Higher Dimensions).

1 Introduction

A *triangulation* of a manifold M is a homeomorphism $H\colon |\mathcal{A}| \to M$, where $\mathcal{A}$ is a simplicial complex, and $|\mathcal{A}|$ is its underlying topological space. If such a homeomorphism exists, we

[1] Arijit Ghosh is supported by Ramanujan Fellowship (No. SB/S2/RJN-064/2015). Part of this work was done when Arijit Ghosh was a Researcher at Max-Planck-Institute for Informatics, Germany supported by the IndoGerman Max Planck Center for Computer Science (IMPECS).

34th International Symposium on Computational Geometry (SoCG 2018).
Editors: Bettina Speckmann and Csaba D. Tóth; Article No. 9; pp. 9:1–9:14


Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

say that $\mathcal{A}$ *triangulates* M.

The purpose of this paper is to present criteria which ensure that a candidate map H is indeed a homeomorphism. This work is motivated by earlier investigations into the problem of algorithmically constructing a complex that triangulates a given manifold [6, 4]. It complements and is closely related to recent work that investigates a particular natural example of such a map [11].

In the motivating algorithmic setting, we are given a compact manifold M, and a manifold simplicial complex $\mathcal{A}$ is constructed by working locally in Euclidean coordinate charts. Here we lay out criteria, based on local properties that arise naturally in the construction of $\mathcal{A}$, that guarantee that H is a homeomorphism. These criteria, which are summarized in Theorem 16, are based on metric properties of H within "compatible" coordinate charts (Definition 4). The Euclidean metric in the local coordinate chart is central to the analysis, but no explicit metric on $|\mathcal{A}|$ or M is involved, and no explicit assumption of differentiability is required of H or M. However, our only examples that meet the required local criteria are in the differentiable setting. We do not know whether or not our criteria for homeomorphism implicitly imply that M admits a differentiable structure. They do imply that $\mathcal{A}$ is piecewise linear (admits an atlas with piecewise linear transition functions).

Relation to other work

The first demonstrations that differentiable manifolds *can* always be triangulated were constructive. Cairns [9] used coordinate charts to cover the manifold with embeddings of patches of Euclidean triangulations. He showed that if the complexes were sufficiently refined the embedding maps could be perturbed such that they remain embeddings and the images of simplices coincide where patches overlap. A global homeomorphic complex is obtained by identifying simplices with the same image. The technique was later refined and extended [15, 14], but it is not easily adapted to provide triangulation guarantees for complexes constructed by other algorithms.

An alternative approach was developed by Whitney [16] using his result that a manifold can be embedded into Euclidean space. A complex is constructed via a process involving the intersection of the manifold with a fine Cartesian grid in the ambient space, and it is shown that the *closest-point projection map*, which takes a point in the complex to its unique closest point in the manifold, is a homeomorphism. The argument is entwined with this specific construction, and is not easily adapted to other settings.

More recently, Edelsbrunner and Shah [13] defined the restricted Delaunay complex of a subset M of Euclidean space as the nerve of the Voronoi diagram on M when the ambient Euclidean metric is used. They showed that if M is a compact manifold, then the restricted Delaunay complex is homeomorphic to M when the Voronoi diagram satisfies the *closed ball property (cbp)*: Voronoi faces are closed topological balls of the appropriate dimension.

Using the cbp, Amenta and Bern [1] demonstrated a specific sampling density that is sufficient to guarantee that the restricted Delaunay complex triangulates the surface. However, since the complex constructed by their reconstruction algorithm cannot be guaranteed to be exactly the restricted Delaunay complex, a new argument establishing homeomorphism was developed, together with a simplified version of the algorithm [2].

Although it was established in the context of restricted Delaunay triangulations, the cbp is an elegant topological result that applies in more general contexts. For example, it has been used to establish conditions for intrinsic Delaunay triangulations of surfaces [12], and Cheng et al. [10] have indicated how it can be applied for establishing weighted restricted Delaunay triangulations of smooth submanifolds of arbitrary dimension in Euclidean space.

However, the cbp is only applicable to Delaunay-like complexes that can be realized as the nerve of some kind of Voronoi diagram on the manifold. Thus, for example, it does not necessarily apply to the tangential Delaunay complex constructed by Boissonnat and Ghosh [6]. Secondly, even when a Delaunay-like complex is being constructed, it can be difficult to directly verify the properties of the associated Voronoi structure; sampling criteria and conditions on the complex under construction are desired, but may not be easy to obtain from the cbp. A third deficiency of the cbp is that, although it can establish that a complex $\mathcal{A}$ triangulates the manifold M, it does not provide a specific triangulation $H : |\mathcal{A}| \to M$. Such a correspondence allows us to compare *geometric* properties of $|\mathcal{A}|$ and M.

In [6] Whitney's argument was adapted to demonstrate that the closest-point projection maps the tangential Delaunay complex homeomorphically onto the original manifold. The argument is intricate, and like Whitney's, is tailored to the specific complex under consideration. In contrast, the result of [2], especially in the formulation presented by Boissonnat and Oudot [8], guarantees a triangulation of a surface by any complex which satisfies a few easily verifiable properties. However, the argument relies heavily on the the codimension being 1.

If a set of vertices is contained within a sufficiently small neighbourhood on a Riemannian manifold, barycentric coordinates can be defined. So there is a natural map from a Euclidean simplex of the appropriate dimension to the manifold, assuming a correspondence between the vertices of the simplex and those on the manifold. Thus when a complex $\mathcal{A}$ is appropriately defined with vertices on a Riemannian manifold M, there is a natural *barycentric coordinate map* $|\mathcal{A}| \to M$. In [11], conditions are presented which guarantee that this map is a triangulation. Although this map is widely applicable, the intrinsic criteria can be inconvenient, for example, in the setting of Euclidean submanifold reconstruction, and furthermore the closest-point projection map may be preferred for triangulation in that setting.

The argument in [11] is based on a general result [11, Proposition 16] for establishing that a given map is a triangulation of a differentiable manifold. However, the criteria include a bound on the differential of the map, which is not easy to obtain. The analysis required to show that the closest-point projection map meets this bound is formidable, and this motivated the current alternate approach. We have relaxed this constraint to a much more easily verifiable bound on the metric distortion of the map when viewed within a coordinate chart.

The sampling criteria for submanifolds imposed by our main result applied to the closest-point projection map (Theorem 17) are the most relaxed that we are aware of. The result could be applied to improve the sampling guarantees of previous works, e.g., [10, 6].

In outline, the argument we develop here is the same as that of [2], but extends the result to apply to abstract manifolds of arbitrary dimension and submanifolds of $\mathbb{R}^N$ of arbitrary codimension. We first show that the map H is a local homeomorphism, and thus a covering map, provided certain criteria are met. Then injectivity is ensured when we can demonstrate that each component of M contains a point y such that $H^{-1}(y)$ is a single point. A core technical lemma from Whitney [16, Appendix II Lemma 15a] still lies at the heart of our argument.

Overview

The demonstration is developed abstractly without explicitly defining the map H. We assume that it has already been established that the restriction of H to any Euclidean simplex in $|\mathcal{A}|$ is an embedding. This is a nontrivial step that needs to be resolved from the specific properties of a particular choice of H. The criteria for local homeomorphism apply in a common coordinate chart (for $|\mathcal{A}|$ and M), and relate the size and quality of the simplices

with the metric distortion of H, viewed in the coordinate domain. The requirement that leads to injectivity is also expressed in a local coordinate chart; it essentially demands that the images of vertices behave in a natural and expected way.

After demonstrating the main result (Theorem 16), we discuss its application to submanifolds of Euclidean space: Theorem 17 presents criteria that ensure that the closest-point projection map from a complex provides a triangulation of a submanifold. The details can be found in the full version of this work [5].

2 The setting and notation

We assume that $\mathcal{A}$ and M are both compact manifolds of dimension m, without boundary, and we have a map $H\colon |\mathcal{A}| \to M$ that we wish to demonstrate is a homeomorphism. We first show that H is a covering map, i.e., every $y \in M$ admits an open neighbourhood U_y such that $H^{-1}(y)$ is a disjoint union of open sets each of which is mapped homeomorphically onto U_y by H. In our setting it is sufficient to establish that H is a local homeomorphism whose image touches all components of M: Brouwer's invariance of domain then ensures that H is surjective, and, since $|\mathcal{A}|$ is compact, has the covering map property.

▶ **Notation 1** (simplices and stars)**.** In this section, a simplex $\boldsymbol{\sigma}$ will always be a *full simplex*: a closed Euclidean simplex, specified by a set of vertices together with all the points with nonnegative barycentric coordinates. The *relative interior* of $\boldsymbol{\sigma}$ is the topological interior of $\boldsymbol{\sigma}$ considered as a subspace of its affine hull, and is denoted by $\operatorname{relint}(\boldsymbol{\sigma})$. If $\boldsymbol{\sigma}$ is a simplex of $\mathcal{A}$, the subcomplex consisting of all simplices that have $\boldsymbol{\sigma}$ as a face, together with the faces of these simplices, is called the *star* of $\boldsymbol{\sigma}$, denoted by $\underline{\mathrm{St}}(\boldsymbol{\sigma})$; the star of a vertex p is $\underline{\mathrm{St}}(p)$.

We also sometimes use the *open star* of a simplex $\boldsymbol{\sigma} \in \mathcal{C}$. This is the union of the relative interiors of the simplices in $\mathcal{C}$ that have $\boldsymbol{\sigma}$ as a face: $\operatorname{st}(\boldsymbol{\sigma}) = \bigcup_{\boldsymbol{\tau} \supseteq \boldsymbol{\sigma}} \operatorname{relint}(\boldsymbol{\tau})$. It is an open set in $|\mathcal{C}|$.

▶ **Notation 2** (topology)**.** If $A \subseteq \mathbb{R}^n$, then the topological closure, interior, and boundary of A are denoted respectively by $\overline{A}$, $\operatorname{int}(A)$, and $\partial A = \overline{A} \setminus \operatorname{int}(A)$. We denote by $B_{\mathbb{R}^m}(c, r)$ the open ball in $\mathbb{R}^m$ of radius r and centre c.

▶ **Notation 3** (linear algebra)**.** The Euclidean norm of $v \in \mathbb{R}^m$ is denoted by $|v|$, and $\|A\| = \sup_{|x|=1} |Ax|$ denotes the operator norm of the linear operator A.

We will work in local coordinate charts. To any given map $G\colon |\mathcal{C}| \to \mathbb{R}^m$, where $\mathcal{C}$ is a simplicial complex, we associate a piecewise linear map $\widehat{G}$ that agrees with G on the vertices of $\mathcal{C}$, and maps $x \in \boldsymbol{\sigma} \in \mathcal{C}$ to the point with the same barycentric coordinates with respect to the images of the vertices. The map $\widehat{G}$ is called the *secant map* of G with respect to $\mathcal{C}$.

The following definition provides the framework within which we will work (see the diagram on page 5).

▶ **Definition 4** (compatible atlases)**.** We say that $|\mathcal{A}|$ and M have *compatible atlases* for $H\colon |\mathcal{A}| \to M$ if:

1. There is a coordinate atlas $\{(U_p, \phi_p)\}_{p \in \mathcal{P}}$ for M, where the index set $\mathcal{P}$ is the set of vertices of $\mathcal{A}$ and each set U_p is connected.
2. For each $p \in \mathcal{P}$, $H(|\underline{\mathrm{St}}(p)|) \subset U_p$. Also, the secant map of $\Phi_p = \phi_p \circ H|_{|\underline{\mathrm{St}}(p)|}$ defines a piecewise linear embedding of $|\underline{\mathrm{St}}(p)|$ into $\mathbb{R}^m$. We denote this secant map by $\widehat{\Phi}_p$. By definition, $\widehat{\Phi}_p$ preserves the barycentric coordinates within each simplex, and thus the collection $\{(\underline{\mathrm{St}}(p), \widehat{\Phi}_p)\}_{p \in \mathcal{P}}$ provides a piecewise linear atlas for $\mathcal{A}$.

▶ **Observation 5.** The requirement in Definition 4 that the local patches U_p be connected implies that on each connected component M' of M, there is a $p \in \mathcal{P}$ such that $H(p) \in M'$.

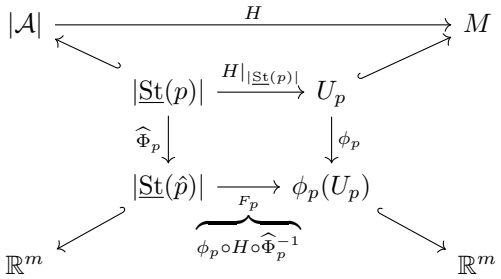
For convenience, we define $\hat{p} = \widehat{\Phi}_p(p)$, so that $\widehat{\Phi}_p(|\underline{\mathrm{St}}(p)|) = |\underline{\mathrm{St}}(\hat{p})|$. We will work within the compatible local coordinate charts. Thus we are studying a map of the form

$$F_p \colon |\underline{\mathrm{St}}(\hat{p})| \subset \mathbb{R}^m \to \mathbb{R}^m,$$

where

$$F_p = \phi_p \circ H \circ \widehat{\Phi}_p^{-1}, \tag{1}$$

as shown in the following diagram (recall M and $\mathcal{A}$ are of dimension m):



We will focus on the map F_p, which can be considered as a local realisation of $H|_{|\underline{\mathrm{St}}(p)|}$. By construction, F_p leaves the vertices of $\underline{\mathrm{St}}(\hat{p})$ fixed: if $q \in \mathbb{R}^m$ is a vertex of $\underline{\mathrm{St}}(\hat{p})$, then $F_p(q) = q$, since $\widehat{\Phi}_p$ coincides with $\phi_p \circ H$ on vertices.

3 Local homeomorphism

Our goal is to ensure that there is some open $V_p \subset |\underline{\mathrm{St}}(\hat{p})|$ such that $F_p|_{V_p}$ is an embedding and that the sets $\widetilde{V}_p = \widehat{\Phi}_p^{-1}(V_p)$ are sufficiently large to cover $|\mathcal{A}|$. This will imply that H is a local homeomorphism. Indeed, if V_p is embedded by F_p, then $\widetilde{V}_p$ is embedded by $H|_{\widetilde{V}_p} = \phi_p^{-1} \circ F_p \circ \widehat{\Phi}_p|_{\widetilde{V}_p}$, since ϕ_p and $\widehat{\Phi}_p$ are both embeddings. Since $|\mathcal{A}|$ is compact, Brouwer's invariance of domain, together with Observation 5, implies that H is surjective, and a covering map. It will only remain to ensure that H is also injective.

We assume that we are given (i.e., we can establish by context-dependent means) a couple of properties of F_p. We require that it be *simplexwise positive*, which means that it is continuous and its restriction to any m-simplex in $\underline{\mathrm{St}}(\hat{p})$ is an orientation preserving topological embedding. As discussed in [5, Appendix A], we can use degree theory to talk about orientation-preserving maps even if the maps are not differentiable. The other requirement we have for F_p is that when it is restricted to an m-simplex it does not distort distances very much, as discussed below.

The local homeomorphism demonstration is based on Lemma 6 below, which is a particular case of an observation made by Whitney [16, Appendix II Lemma 15a]. Whitney demonstrated a more general result from elementary first principles. The proof we give here is roughly the same as Whitney's, except that we exploit elementary degree theory, as discussed in [5, Appendix A], in order to avoid the differentiability assumptions Whitney made.

In the statement of the lemma, $\mathcal{C}^{m-1}$ refers to the $(m-1)$-*skeleton* of the complex $\mathcal{C}$: the subcomplex consisting of simplices of dimension less than or equal to $m-1$. When $|\mathcal{C}|$ is a manifold with boundary, as in the lemma, then $\partial\mathcal{C}$ is the subcomplex containing all $(m-1)$-simplices that are the face of a single m-simplex, together with the faces of these simplices.

▶ **Lemma 6** (simplexwise positive embedding). *Assume $\mathcal{C}$ is an oriented m-manifold finite simplicial complex with boundary embedded in $\mathbb{R}^m$. Let $F\colon |\mathcal{C}| \to \mathbb{R}^m$ be simplexwise positive in $\mathcal{C}$. Suppose $V \subset |\mathcal{C}|$ is a connected open set such that $F(V) \cap F(|\partial\mathcal{C}|) = \emptyset$. If there is a $y \in F(V) \setminus F(|\mathcal{C}^{m-1}|)$ such that $F^{-1}(y)$ is a single point, then the restriction of F to V is a topological embedding.*

Proof. Notice that the topological boundary of $|\mathcal{C}| \subset \mathbb{R}^m$ is equal to the underlying space of the boundary complex (see, e.g., [3, Lemmas 3.6, 3.7]): $\partial|\mathcal{C}| = |\partial\mathcal{C}|$. Let $\Omega = |\mathcal{C}| \setminus |\partial\mathcal{C}|$. Since F is simplexwise positive, and $F(V)$ lies within a connected component of $\mathbb{R}^m \setminus F(\partial\Omega)$, the fact that $F^{-1}(y)$ is a single point implies that $F^{-1}(w)$ is a single point for any $w \in F(V)\setminus F(|C^{m-1}|)$ (see [5, Lemma 49]). We need to show that F is also injective on $V \cap |\mathcal{C}^{m-1}|$.

Let $\boldsymbol{\sigma} \in \mathcal{C}^{m-1} \setminus \partial\mathcal{C}$, and observe that the open star $\mathrm{st}(\boldsymbol{\sigma})$ (Notation 1) is open in $\mathbb{R}^m$. We now show that $F(\mathrm{st}(\boldsymbol{\sigma}))$ is open. Suppose $x \in \mathrm{relint}(\boldsymbol{\tau})$ for some $\boldsymbol{\tau} \in \mathcal{C} \setminus \partial\mathcal{C}$. Since F is injective when restricted to any simplex, we can find a sufficiently small open (in $\mathbb{R}^m$) neighbourhood U of $F(x)$ such that $U \cap F(\partial\mathrm{st}(\boldsymbol{\tau})) = \emptyset$. Since the closure of the open star is equal to the underlying space of our usual star: $\overline{\mathrm{st}(\boldsymbol{\tau})} = |\underline{\mathrm{St}}(\boldsymbol{\tau})|$, By [5, Lemma 49], every point in $U \setminus F(|\underline{\mathrm{St}}(\boldsymbol{\tau})^{m-1}|)$ has the same number of points in its preimage. By the injectivity of F restricted to m-simplices, this number must be greater than zero for points near $F(x)$. It follows that $U \subseteq F(\mathrm{st}(\boldsymbol{\tau}))$.

If $x \in \mathrm{st}(\boldsymbol{\sigma})$, then $x \in \mathrm{relint}(\boldsymbol{\tau})$ for some $\boldsymbol{\tau} \in \mathcal{C} \setminus \partial\mathcal{C}$ that has $\boldsymbol{\sigma}$ as a face. Since $\mathrm{st}(\boldsymbol{\tau}) \subseteq \mathrm{st}(\boldsymbol{\sigma})$, we have $U \subseteq F(\mathrm{st}(\boldsymbol{\sigma}))$, and we conclude that $F(\mathrm{st}(\boldsymbol{\sigma}))$ is open.

Now, to see that F is injective on $|\mathcal{C}^{m-1}| \cap V$, suppose to the contrary that $w, z \in |\mathcal{C}^{m-1}| \cap V$ are two distinct points such that $F(w) = F(z)$. Since F is injective on each simplex, there are distinct simplices $\boldsymbol{\sigma}, \boldsymbol{\tau}$ such that $w \in \mathrm{relint}(\boldsymbol{\sigma})$ and $z \in \mathrm{relint}(\boldsymbol{\tau})$. So there is an open neighbourhood U of $F(w) = F(z)$ that is contained in $F(\mathrm{st}(\boldsymbol{\sigma})) \cap F(\mathrm{st}(\boldsymbol{\tau}))$.

We must have $\mathrm{st}(\boldsymbol{\sigma}) \cap \mathrm{st}(\boldsymbol{\tau}) = \emptyset$, because if $x \in \mathrm{st}(\boldsymbol{\sigma}) \cap \mathrm{st}(\boldsymbol{\tau})$, then $x \in \mathrm{relint}(\boldsymbol{\mu})$ for some $\boldsymbol{\mu}$ that has both $\boldsymbol{\sigma}$ and $\boldsymbol{\tau}$ as faces. But this means that both w and z belong to $\boldsymbol{\mu}$, contradicting the injectivity of $F|_{\boldsymbol{\mu}}$. It follows that points in the nonempty set $U \setminus |\mathcal{C}^{m-1}|$ have at least two points in their preimage, a contradiction. Thus $F|_V$ is injective, and it follows from Brouwer's invariance of domain that $F|_V$ is an embedding. ◀

Our strategy for employing Lemma 6 is to demand that the restriction of F_p to any m-simplex has low metric distortion, and use this fact to ensure that the image of $V_p \subset |\underline{\mathrm{St}}(\hat{p})|$ is not intersected by the image of the boundary of $|\underline{\mathrm{St}}(\hat{p})|$, i.e., we will establish that $F_p(V_p) \cap F_p(|\partial\underline{\mathrm{St}}(\hat{p})|) = \emptyset$. We need to also establish that there is a point y in $F_p(V_p) \setminus F_p(|\underline{\mathrm{St}}(\hat{p})^{m-1}|)$ such that $F^{-1}(y)$ is a single point. The metric distortion bound will help us here as well.

▶ **Definition 7** (ξ-distortion map). A map $F\colon U \subset \mathbb{R}^m \to \mathbb{R}^m$ is a *ξ-distortion map* if for all $x, y \in U$ we have

$$\big||F(x) - F(y)| - |x - y|\big| \leq \xi\,|x - y|\,. \tag{2}$$

We are interested in ξ-distortion maps with small ξ. Equation (2) can be equivalently written

$$(1 - \xi)\,|x - y| \leq |F(x) - F(y)| \leq (1 + \xi)\,|x - y|\,,$$

and it is clear that when $\xi < 1$, a ξ-distortion map is a bi-Lipschitz map. For our purposes the metric distortion constant ξ is more convenient than a bi-Lipschitz constant. It is easy

to show that if F is a ξ-distortion map, with $\xi < 1$, then F is a homeomorphism onto its image, and F^{-1} is a $\frac{\xi}{1-\xi}$-distortion map (see [5, Lemma 19(1)]).

Assuming that $F_p|_{\boldsymbol{\sigma}}$ is a ξ-distortion map for each m-simplex $\boldsymbol{\sigma} \in \underline{\mathrm{St}}(\hat{p})$, we can bound how much it displaces points. Specifically, for any point $x \in |\underline{\mathrm{St}}(\hat{p})|$, we will bound $|x - F(x)|$. We exploit the fact that the $m+1$ vertices of $\boldsymbol{\sigma}$ remain fixed, and use *trilateration*, i.e., we use the estimates of the distances to the fixed vertices to estimate the location of $F(x)$. Here, the quality of the simplex comes into play.

▶ **Notation 8** (simplex quality). If p is a vertex of $\boldsymbol{\sigma}$, the *altitude* of p is the distance from p to the opposing facet of $\boldsymbol{\sigma}$ and is denoted $a_p(\boldsymbol{\sigma})$. The *thickness* of $\boldsymbol{\sigma}$, denoted $t(\boldsymbol{\sigma})$ (or just t if there is no risk of confusion) is given by $\frac{a}{mL}$, where $a = a(\boldsymbol{\sigma})$ is the smallest altitude of $\boldsymbol{\sigma}$, and $L = L(\boldsymbol{\sigma})$ is the length of the longest edge. We set $t(\boldsymbol{\sigma}) = 1$ if $\boldsymbol{\sigma}$ has dimension 0.

▶ **Lemma 9** (trilateration). *Suppose $\boldsymbol{\sigma} \subset \mathbb{R}^m$ is an m-simplex, and $F\colon \boldsymbol{\sigma} \to \mathbb{R}^m$ is a ξ-distortion map that leaves the vertices of $\boldsymbol{\sigma}$ fixed. If $\xi \leq 1$, then for any $x \in \boldsymbol{\sigma}$,*

$$|x - F(x)| \leq \frac{3\xi L}{t},$$

where L is the length of the longest edge of $\boldsymbol{\sigma}$, and t is its thickness.

Proof. Let $\{p_0, \ldots, p_m\}$ be the vertices of $\boldsymbol{\sigma}$. For $x \in \boldsymbol{\sigma}$, let $\tilde{x} = F(x)$.

We choose p_0 as the origin, and observe that

$$p_i{}^{\mathsf{T}} x = \frac{1}{2}\left(|x|^2 + |p_i|^2 - |x - p_i|^2\right), \tag{3}$$

which we write in matrix form as $P^{\mathsf{T}} x = b$, where P is the $m \times m$ matrix whose i-th column is p_i, and b is the vector whose i-th component is given by the right-hand side of (3). Similarly, we have $P^{\mathsf{T}} \tilde{x} = \tilde{b}$ with the obvious definition of $\tilde{b}$. Then

$$\tilde{x} - x = (P^{\mathsf{T}})^{-1}(\tilde{b} - b).$$

Since $F(p_0) = p_0 = 0$, we have $||\tilde{x}| - |x|| \leq \xi\,|x|$, and so

$$||\tilde{x}|^2 - |x|^2| \leq \xi(2+\xi)\,|x|^2 \leq 3\xi L^2.$$

Similarly, $||x - p_i|^2 - |\tilde{x} - p_i|^2| < 3\xi L^2$. Thus $|\tilde{b}_i - b_i| \leq 3\xi L^2$, and $|\tilde{b} - b| \leq 3\sqrt{m}\xi L^2$.

By [3, Lemma 2.4] we have $\left\|(P^{\mathsf{T}})^{-1}\right\| \leq (\sqrt{m}tL)^{-1}$, and the stated bound follows. ◀

We define V_p to be the open set obtained by homothetically "shrinking" $|\underline{\mathrm{St}}(\hat{p})|$ such that it is just large enough to contain the barycentres of the simplices that have $\hat{p}$ as a vertex (see Figure 1). To be more specific we define V_p to be the open set consisting of the points in $|\underline{\mathrm{St}}(\hat{p})|$ whose barycentric coordinate with respect to $\hat{p}$ is strictly larger than $\frac{1}{m+1} - \delta$, where $\delta > 0$ is arbitrarily small. Since the barycentric coordinates in each m-simplex sum to 1, and the piecewise linear maps $\widehat{\Phi}_p$ preserve barycentric coordinates, this ensures that the sets $\widehat{\Phi}_p^{-1}(V_p)$ cover $|\mathcal{A}|$.

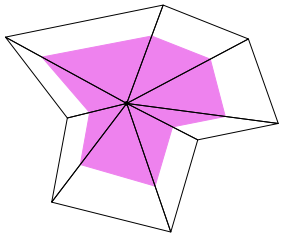


Figure 1 The open set V_p.

We assume that F_p is a ξ-distortion map on each simplex. The idea is to show that F_p is an embedding on V_p. In order to employ the simplexwise positive embedding lemma (Lemma 6), we need to establish that there is a point in $V_p \setminus |\underline{\mathrm{St}}(\hat{p})^{m-1}|$ that is not mapped to the image of any other point in $|\underline{\mathrm{St}}(\hat{p})|$. We choose the barycentre of a simplex for this purpose. We say that a simplicial complex is a *pure* m-dimensional simplicial complex if every simplex is the face of an m-simplex.

▶ **Lemma 10** (a point covered once). *Suppose $\mathcal{C}$ is a pure m-dimensional finite simplicial complex embedded in $\mathbb{R}^m$, and that for each $\boldsymbol{\sigma} \in \mathcal{C}$ we have $t(\boldsymbol{\sigma}) \geq t_0$. Let $\boldsymbol{\sigma} \in \mathcal{C}$ be an m-simplex with the largest diameter, i.e., $L(\boldsymbol{\sigma}) \geq L(\boldsymbol{\tau})$ for all $\boldsymbol{\tau} \in \mathcal{C}$, and let b be the barycentre of $\boldsymbol{\sigma}$. If $F\colon |\mathcal{C}| \to \mathbb{R}^m$ leaves the vertices of $\mathcal{C}$ fixed, and its restriction to any m-simplex in $\mathcal{C}$ is a ξ-distortion map with*

$$\xi \leq \frac{1}{6}\frac{m}{m+1}t_0^2, \tag{4}$$

then $F^{-1}(F(b)) = \{b\}$.

Proof. Since $\xi < 1$, the restriction of F to $\boldsymbol{\sigma}$ is injective. Suppose $x \in |\mathcal{C}|$ is such that $F(x) = F(b)$, but $x \neq b$. Then x belongs to some m-simplex $\boldsymbol{\tau} \in \mathcal{C}$ different from $\boldsymbol{\sigma}$. Since the distance from b to the boundary of $\boldsymbol{\sigma}$ is

$$\frac{a(\boldsymbol{\sigma})}{m+1} = \frac{mt(\boldsymbol{\sigma})L(\boldsymbol{\sigma})}{m+1},$$

it follows that $|x-b| > mt(\boldsymbol{\sigma})L(\boldsymbol{\sigma})/(m+1)$. But using Lemma 9 and the constraint (4) we arrive at a contradiction:

$$|x-b| \leq |b-F(b)| + |x-F(x)| \leq \frac{3\xi L(\boldsymbol{\sigma})}{t(\boldsymbol{\sigma})} + \frac{3\xi L(\boldsymbol{\tau})}{t(\boldsymbol{\tau})} \leq \frac{mt(\boldsymbol{\sigma})L(\boldsymbol{\sigma})}{m+1}. \qquad \blacktriangleleft$$

Now we also need to ensure that $F_p(V_p) \cap F_p(|\partial\underline{\mathrm{St}}(\hat{p})|) = \emptyset$. Here we will explicitly use the assumption that $\underline{\mathrm{St}}(\hat{p})$ is $\underline{\mathrm{St}}(\hat{p})$. We say that $\underline{\mathrm{St}}(\hat{p})$ is a *full star* if its underlying space is an m-manifold with boundary and $\hat{p}$ does not belong to $\partial\underline{\mathrm{St}}(\hat{p})$.

▶ **Lemma 11** (barycentric boundary separation). *Suppose $\underline{\mathrm{St}}(\hat{p})$ is a full m-dimensional star embedded in $\mathbb{R}^m$. Let $a_0 = \min_{\boldsymbol{\sigma}\in\underline{\mathrm{St}}(\hat{p})} a_{\hat{p}}(\boldsymbol{\sigma})$ be the smallest altitude of $\hat{p}$ in the m-simplices in $\underline{\mathrm{St}}(\hat{p})$. Suppose $x \in \boldsymbol{\sigma} \in \underline{\mathrm{St}}(\hat{p})$, where $\boldsymbol{\sigma}$ is an m-simplex, and $\lambda_{\boldsymbol{\sigma},\hat{p}}(x)$, the barycentric coordinate of x with respect to $\hat{p}$ in $\boldsymbol{\sigma}$, satisfies $\lambda_{\boldsymbol{\sigma},\hat{p}}(x) \geq \alpha$. Then $d_{\mathbb{R}^m}(x, |\partial\underline{\mathrm{St}}(\hat{p})|) \geq \alpha a_0$.*

If t_0 is a lower bound on the thicknesses of the simplices in $\underline{\mathrm{St}}(\hat{p})$, and s_0 is a lower bound on their diameters, then $d_{\mathbb{R}^m}(x, |\partial\underline{\mathrm{St}}(\hat{p})|) \geq \alpha m t_0 s_0$.

Proof. Since we are interested in the distance to the boundary, consider a point $y \in |\partial\underline{\mathrm{St}}(\hat{p})|$ such that the segment $[x,y]$ lies in $|\underline{\mathrm{St}}(\hat{p})|$. The segment passes through a sequence of m-simplices, $\boldsymbol{\sigma}_0 = \boldsymbol{\sigma}, \boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_n$, that partition it into subsegments $[x_i, y_i] \subset \boldsymbol{\sigma}_i$ with $x_0 = x$, $y_n = y$ and $x_i = y_{i-1}$ for all $i \in \{1, \ldots, n\}$.

Observe that $\lambda_{\boldsymbol{\sigma}_i,\hat{p}}(x_i) = \lambda_{\boldsymbol{\sigma}_{i-1},\hat{p}}(y_{i-1})$, and that

$$|x_i - y_i| \geq a_{\hat{p}}(\boldsymbol{\sigma}_i)|\lambda_{\boldsymbol{\sigma}_i,\hat{p}}(x_i) - \lambda_{\boldsymbol{\sigma}_i,\hat{p}}(y_i)|.$$

Thus

$$\begin{aligned}|x-y| = \sum_{i=0}^{n}|x_i-y_i| &\geq \sum_{i=0}^{n} a_{\hat{p}}(\boldsymbol{\sigma}_i)|\lambda_{\boldsymbol{\sigma}_i,\hat{p}}(x_i) - \lambda_{\boldsymbol{\sigma}_i,\hat{p}}(y_i)| \geq a_0\sum_{i=0}^{n}(\lambda_{\boldsymbol{\sigma}_i,\hat{p}}(x_i) - \lambda_{\boldsymbol{\sigma}_i,\hat{p}}(y_i)) \\ &= a_0(\lambda_{\boldsymbol{\sigma},\hat{p}}(x) - \lambda_{\boldsymbol{\sigma}_n,\hat{p}}(y)) = a_0\lambda_{\boldsymbol{\sigma},\hat{p}}(x) \geq a_0\alpha.\end{aligned}$$

From the definition of thickness we find that $a_0 \geq t_0 m s_0$, yielding the second statement of the lemma. $\blacktriangleleft$

Lemma 11 allows us to quantify the distortion bound that we need to ensure that the boundary of $\underline{\mathrm{St}}(\hat{p})$ does not get mapped by F_p into the image of the open set V_p. The argument is the same as for Lemma 10, but there we were only concerned with the barycentre of the largest simplex, so the relative sizes of the simplices were not relevant as they are here (compare the bounds (4) and (5)).

▶ **Lemma 12** (boundary separation for V_p). *Suppose $\underline{\mathrm{St}}(\hat{p})$ is a full star embedded in $\mathbb{R}^m$, and every m-simplex $\boldsymbol{\sigma}$ in $\underline{\mathrm{St}}(\hat{p})$ satisfies $s_0 \leq L(\boldsymbol{\sigma}) \leq L_0$, and $t(\boldsymbol{\sigma}) \geq t_0$. If the restriction of F_p to any m-simplex in $\underline{\mathrm{St}}(\hat{p})$ is a ξ-distortion map, with*

$$\xi < \frac{1}{6}\frac{m}{m+1}\frac{s_0}{L_0}t_0^2, \tag{5}$$

then $F_p(V_p) \cap F_p(|\partial\underline{\mathrm{St}}(\hat{p})|) = \emptyset$, where V_p is the set of points with barycentric coordinate with respect to $\hat{p}$ in a containing m-simplex strictly greater than $\frac{1}{m+1} - \delta$, with $\delta > 0$ an arbitrary, suffiently small parameter.

Proof. If $x \in \boldsymbol{\sigma} \in \underline{\mathrm{St}}(\hat{p})$ has barycentric coordinate with respect to $\hat{p}$ larger than $\frac{1}{m+1} - \delta$, and $y \in \boldsymbol{\tau} \in \partial\underline{\mathrm{St}}(\hat{p})$, then Lemmas 9 and 11 ensure that $F_p(x) \neq F_p(y)$ provided

$$\frac{3\xi L(\boldsymbol{\sigma})}{t(\boldsymbol{\sigma})} + \frac{3\xi L(\boldsymbol{\tau})}{t(\boldsymbol{\tau})} \leq \left(\frac{1}{m+1} - \delta\right) m s_0 t_0,$$

which is satisfied by (5) when $\delta > 0$ satisfies

$$\delta \leq \frac{1}{m+1} - \frac{6L_0\xi}{m s_0 t_0^2}.$$ ◀

When inequality (5) (and therefore also inequality (4)) is satisfied, we can employ the embedding lemma (Lemma 6) to guarantee that V_p is embedded:

▶ **Lemma 13** (local homeomorphism). *Suppose M is an m-manifold and $\mathcal{A}$ is a simplicial complex with vertex set $\mathcal{P}$. A map $H\colon |\mathcal{A}| \to M$ is a covering map if the following criteria are satisfied:*

1. **manifold complex** *$\mathcal{A}$ is a compact m-manifold complex (without boundary).*
2. **compatible atlases** *There are compatible atlases for H* (Definition 4).
3. **simplex quality** *For each $p \in \mathcal{P}$, every simplex $\boldsymbol{\sigma} \in \underline{\mathrm{St}}(\hat{p}) = \widehat{\Phi}_p(\underline{\mathrm{St}}(p))$ satisfies $s_0 \leq L(\boldsymbol{\sigma}) \leq L_0$ and $t(\boldsymbol{\sigma}) \geq t_0$* (Notation 8).
4. **distortion control** *For each $p \in \mathcal{P}$, the map*

   $$F_p = \phi_p \circ H \circ \widehat{\Phi}_p^{-1}\colon |\underline{\mathrm{St}}(\hat{p})| \to \mathbb{R}^m,$$

   when restricted to any m-simplex in $\underline{\mathrm{St}}(\hat{p})$, is an orientation-preserving ξ-distortion map (Definition 7) *with*

   $$\xi < \frac{m s_0 t_0^2}{6(m+1)L_0}.$$ ◀

4 Injectivity

Having established that H is a covering map, to ensure that H is injective it suffices to demonstrate that on each component of M there is a point with only a single point in its preimage. Injectivity follows since the number of points in the preimage is locally constant for covering maps.

Since each simplex is embedded by H, it is sufficient to show that for each vertex $q \in \mathcal{P}$, if $H(q) \in H(\boldsymbol{\sigma})$, then q is a vertex of $\boldsymbol{\sigma}$. This ensures that $H^{-1}(H(q)) = \{q\}$, and by Observation 5 each component of M must contain the image of a vertex.

In practice, we typically don't obtain this condition directly. The complex $\mathcal{A}$ is *constructed* by means of the local coordinate patches $\underline{\mathrm{St}}(\hat{p})$, and it is with respect to these patches that the vertices behave well.

▶ **Definition 14** (vertex sanity). If $H\colon |\mathcal{A}| \to M$ has compatible atlases (Definition 4), then H exibits *vertex sanity* if: for all vertices $p, q \in \mathcal{P}$, if $\phi_p \circ H(q) \in |\underline{\mathrm{St}}(\hat{p})| = \widehat{\Phi}_p(|\underline{\mathrm{St}}(p)|)$, then q is a vertex of $\underline{\mathrm{St}}(p)$.

Together with the distortion bounds that are imposed on F_p, Definition 14 ensures that the image of a vertex cannot lie in the image of a simplex to which it does not belong:

▶ **Lemma 15** (injectivity). *If $H\colon |\mathcal{A}| \to M$ satisfies the hypotheses of Lemma 13 as well as Definition 14, then H is injective, and therefore a homeomorphism.*

Proof. Towards a contradiction, suppose that $H(q) \in H(\boldsymbol{\sigma})$ and that q is not a vertex of the m-simplex $\boldsymbol{\sigma}$. This means there is some $x \in \boldsymbol{\sigma}$ such that $H(x) = H(q)$. Let p be a vertex of $\boldsymbol{\sigma}$. The vertex sanity hypothesis (Definition 14) implies that $\phi_p \circ H(q)$ must be either outside of $|\underline{\mathrm{St}}(\hat{p})|$, or belong to its boundary. Thus Lemmas 11 and 9, and the bound on ξ from Lemma 13(3) imply that the barycentric coordinate of x with respect to p must be smaller than $\frac{1}{m+1}$: Let $\hat{x} = \widehat{\Phi}_p(x)$, and $\hat{\boldsymbol{\sigma}} = \widehat{\Phi}_p(\boldsymbol{\sigma})$. Lemma 9 says that

$$|F_p(\hat{x}) - \hat{x}| \leq \frac{3\xi L_0}{t_0} < \frac{m s_0 t_0}{2(m+1)} \leq \frac{a_0}{2(m+1)},$$

where a_0 is a lower bound on the altitudes of $\hat{p}$, as in Lemma 11. Since $F_p(\hat{x}) = \phi_p \circ H(x)$ is at least as far away from $\hat{x}$ as $\partial\underline{\mathrm{St}}(\hat{p})$, Lemma 11 implies that the barycentric coordinate of $\hat{x} \in \hat{\boldsymbol{\sigma}}$ with respect to $\hat{p}$ must be no larger than $\frac{1}{2(m+1)}$. Since $\widehat{\Phi}_p$ preserves barycentric coordinates, and the argument works for any vertex p of $\boldsymbol{\sigma}$, we conclude that all the barycentric coordinates of x in $\boldsymbol{\sigma}$ are strictly less than $\frac{1}{m+1}$. We have reached a contradiction with the fact that the barycentric coordinates of x must sum to 1. ◀

5 Main result

To recap, Lemmas 13 and 15 yield the following triangulation result. In the bound on ξ from Lemma 13(3), we replace the factor $\frac{m}{m+1}$ with $\frac{1}{2}$, the lower bound attained when $m = 1$.

▶ **Theorem 16** (triangulation). *Suppose M is an m-manifold, and $\mathcal{A}$ is a simplicial complex with vertex set $\mathcal{P}$. A map $H\colon |\mathcal{A}| \to M$ is a homeomorphism if the following criteria are satisfied:*

1. **manifold complex** *$\mathcal{A}$ is a compact m-manifold complex (without boundary).*
2. **compatible atlases** *There are compatible atlases for H* (Definition 4)*:*

$$\{(\underline{\mathrm{St}}(p), \widehat{\Phi}_p)\}_{p\in\mathcal{P}}, \quad \underline{\mathrm{St}}(p) \subset \mathcal{A}, \qquad \text{and} \qquad \{(U_p, \phi_p)\}_{p\in\mathcal{P}}, \quad U_p \subset M.$$

3. **simplex quality** *For each $p \in \mathcal{P}$, every simplex $\boldsymbol{\sigma} \in \underline{\mathrm{St}}(\hat{p}) = \widehat{\Phi}_p(\underline{\mathrm{St}}(p))$ satisfies $s_0 \leq L(\boldsymbol{\sigma}) \leq L_0$ and $t(\boldsymbol{\sigma}) \geq t_0$* (Notation 8)*.*
4. **distortion control** *For each $p \in \mathcal{P}$, the map*

$$F_p = \phi_p \circ H \circ \widehat{\Phi}_p^{-1}\colon |\underline{\mathrm{St}}(\hat{p})| \to \mathbb{R}^m,$$

when restricted to any m-simplex in $\underline{\mathrm{St}}(\hat{p})$, is an orientation-preserving ξ-distortion map (Definition 7) *with*

$$\xi < \frac{s_0 t_0^2}{12 L_0}.$$

5. **vertex sanity** *For all vertices $p, q \in \mathcal{P}$, if $\phi_p \circ H(q) \in |\underline{\mathrm{St}}(\hat{p})|$, then q is a vertex of $\underline{\mathrm{St}}(p)$.*

◀

▶ Remark. The constants L_0, s_0, and t_0 that constrain the simplices in the local complex $\underline{\mathrm{St}}(\hat{p})$, and the metric distortion of F_p in Theorem 16 can be considered to be local, i.e., they may depend on $p \in \mathcal{P}$. This result applies for any dimension m, but for $m \leq 2$, triangulation criteria already exist which demand neither a lower bound on the size (s_0), nor on the quality (t_0) of the simplices [2, 8].

Application: submanifolds of Euclidean space

As a specific application of Theorem 16, we consider a smooth (or at least C^2) compact m-dimensional submanifold of Euclidean space: $M \subset \mathbb{R}^N$. A simplicial complex $\mathcal{A}$ is built whose vertices are a finite set $\mathcal{P}$ sampled from the manifold: $\mathcal{P} \subset M$. The motivating model for this setting is the tangential Delaunay complex [6]. In that case $\mathcal{A}$ is constructed as a subcomplex of a weighted Delaunay triangulation of $\mathcal{P}$ in the ambient space $\mathbb{R}^N$, so it is necessarily embedded. However, in general we do not need to assume *a priori* that $\mathcal{A}$ is embedded in $\mathbb{R}^N$. (This does not force us to consider $\mathcal{A}$ to be abstract in the combinatorial sense. In particular, the simplices are Euclidean simplices, not just sets of vertices.) Instead, we assume only that the embedding of the vertex set $\mathcal{P} \hookrightarrow \mathbb{R}^N$ defines an *immersion* $\iota\colon |\mathcal{A}| \to \mathbb{R}^N$. By this we mean that for any vertex $p \in \mathcal{P}$ we have that the restriction of ι to $|\underline{\mathrm{St}}(p)|$ is an embedding.

At each point $x \in M$, the tangent space $T_xM \subset T_x\mathbb{R}^N$ is naturally viewed as an m-dimensional affine flat in $\mathbb{R}^N$, with the vector-space structure defined by taking the distinguished point x as the origin. The maps involved in Theorem 16 will be defined by projection maps. The coordinate charts are defined using the orthogonal projection $\mathrm{pr}_{T_pM}\colon \mathbb{R}^N \to T_pM$. As discussed in [5, Section 4.3], for a sufficiently small neighbourhood $U_p \subset M$, we obtain an embedding

$$\phi_p = \mathrm{pr}_{T_pM}|_{U_p}\colon U_p \subset M \to T_pM \cong \mathbb{R}^m,$$

which will define our coordinate maps for M.

For the map $H\colon |\mathcal{A}| \to M$, we employ the closest point projection map: The *medial axis*, $\mathrm{ax}(M)$ is the set of points $x \in \mathbb{R}^N$ that have more than one closest point on M, and $\overline{\mathrm{ax}}(M)$ is its closure. The open set $U_M = \mathbb{R}^N \setminus \overline{\mathrm{ax}}(M)$ contains M and each point in it has a unique closest point on M, so the *closest-point projection map*, $\mathrm{pr}_M\colon U_M \to M$, is well-defined (see, e.g., [5, Section 4.1]). We define $H = \mathrm{pr}_M \circ \iota$.

The *local feature size* is the function $\mathrm{lfs}\colon M \to \mathbb{R}_{\geq 0}$ defined by $\mathrm{lfs}(x) = d_{\mathbb{R}^N}(x, \overline{\mathrm{ax}}(M))$. It is easily verified that this function is continuous. It plays an important role as a sizing function that governs the density of sample points (vertices of $\mathcal{A}$) that are required to construct a triangulation.

As demanded by Definition 4, for each $p \in \mathcal{P}$ the coordinate map $\widehat{\Phi}_p$ for $\mathcal{A}$ is the secant map of $\phi_p \circ H$ restricted to $|\underline{\mathrm{St}}(p)|$, and since pr_{T_pM} is already a linear map, and pr_M is the identity on the vertices, this means $\widehat{\Phi}_p = \mathrm{pr}_{T_pM} \circ \iota|_{|\underline{\mathrm{St}}(p)|}$ (recall that the secant map $\widehat{\Phi}_p$ is defined by the action of $\phi_p \circ H|_{|\underline{\mathrm{St}}(p)|}$ on the vertices alone).

In order to employ Theorem 16, we need to analyze the projection maps to obtain metric distortion bounds on the restriction to m-simplices. For the projection onto the tangent spaces, this analysis [5, Section 4.3] yields bounds on the size of the simplices and coordinate neighbourhoods such that the compatible atlas criterion will be automatically satisfied. Thus the compatible atlas criterion does not appear explicitly in the statement of Theorem 17; it has been subsumed by the proof.

The metric distortion of pr_M restricted to an m-simplex is analyzed in [5, Section 4.4], using recent bounds on the angle between nearby tangent spaces [7]. The distortion bounds on the different projection maps yield a distortion bound on their composition, F_p. In order to obtain a bound in this fashion, it is necessary choose a metric on M. We employed the metric of the ambient space $\mathbb{R}^N$ restricted to M, rather than the intrinsic metric of geodesic distances.

The simplex quality and sampling criteria of Theorem 17(b) are tailored to ensure that the metric distortion criterion of Theorem 16(4) is met. Thus an explicit metric distortion condition does not appear in the statement of Theorem 17. Some adjustment was also made so that the simplex quality conditions in Theorem 17(b) refer to the ambient simplices of $\mathcal{A}$, rather than the projected simplices in the tangent spaces, as required by Theorem 16.

We then arrive at the following specific incarnation of Theorem 16:

▶ **Theorem 17** (triangulation for submanifolds). *Let $M \subset \mathbb{R}^N$ be a compact C^2-continuous manifold, and $\mathcal{P} \subset M$ a finite set of points such that for each connected component M_c of M, $M_c \cap \mathcal{P} \neq \emptyset$. Suppose that $\mathcal{A}$ is a simplicial complex whose vertices, $\mathcal{A}^0$, are identified with $\mathcal{P}$, by a bijection $\mathcal{A}^0 \to \mathcal{P}$ such that the resulting piecewise linear map $\iota\colon |\mathcal{A}| \to \mathbb{R}^N$ is an immersion, i.e., $\iota|_{|\underline{\mathrm{St}}(p)|}$ is an embedding for each vertex p.*

If:

(a) manifold complex *For each vertex $p \in \mathcal{P}$, the projection $\mathrm{pr}_{T_pM}|_{\iota(|\underline{\mathrm{St}}(p)|)}$ is an embedding and p lies in the interior of $\mathrm{pr}_{T_pM}(\iota(|\underline{\mathrm{St}}(p)|))$.*

(b) simplex quality *There are constants $0 < t_0 \leq 1$, $0 < \mu_0 \leq 1$, and $\epsilon_0 > 0$ such that for each simplex $\boldsymbol{\sigma} \in \iota(\mathcal{A})$, and each vertex $p \in \boldsymbol{\sigma}$,*

$$t(\boldsymbol{\sigma}) \geq t_0, \quad \mu_0\epsilon_0 \operatorname{lfs}(p) \leq L(\boldsymbol{\sigma}) \leq \epsilon_0 \operatorname{lfs}(p), \quad \epsilon_0 \leq \frac{\mu_0^{\frac{1}{2}} t_0^2}{18}.$$

(c) vertex sanity *For any vertices $p, q \in \mathcal{P}$, if $q \in U_p = B_{\mathbb{R}^N}(p, r) \cap M$, where $r = \operatorname{lfs}(p)/15$, then $\mathrm{pr}_{T_pM}(q) \in \mathrm{pr}_{T_pM}(\iota(\underline{\mathrm{St}}(p)))$ if and only if q is a vertex of $\underline{\mathrm{St}}(p)$.*

Then:

1. *ι is an embedding, so the complex $\mathcal{A}$ may be identified with $\iota(\mathcal{A})$.*
2. *The closest-point projection map $\mathrm{pr}_M|_{|\mathcal{A}|}$ is a homeomorphism $|\mathcal{A}| \to M$.*
3. *For any $x \in \boldsymbol{\sigma} \in \mathcal{A}$,*

$$\delta_M(x) = |\check{x} - x| \leq \tfrac{7}{3}\epsilon_0^2 \operatorname{lfs}(\check{x}), \quad \textit{and} \quad \sin\angle(\boldsymbol{\sigma}, T_{\check{x}}) \leq \frac{13\epsilon_0}{4t_0},$$

where $\check{x} = \mathrm{pr}_M(x)$.

Using the reach, which is a global bound on the local feature size, $\operatorname{rch}(M) = \inf_{x \in M} \operatorname{lfs}(x)$, we obtain the following variation of Theorem 17, which is a corollary in the sense that it follows from essentially the same proof, even though it does not follow from the statement of Theorem 17.

▶ **Corollary 18.** *If the conditions* (b) *and* (c) *in Theorem 17 are replaced by*

(b′) *There are constants $0 < t_0 \leq 1$, $0 < \mu_0 \leq 1$, and $\epsilon_0 > 0$ such that for each simplex $\boldsymbol{\sigma} \in \iota(\mathcal{A})$, and each vertex $p \in \boldsymbol{\sigma}$,*

$$t(\boldsymbol{\sigma}) \geq t_0, \quad \mu_0\epsilon_0 \operatorname{rch}(M) \leq L(\boldsymbol{\sigma}) \leq \epsilon_0 \operatorname{rch}(M), \quad \epsilon_0 \leq \frac{\mu_0^{\frac{1}{2}} t_0^2}{16}.$$

(c′) *For any vertices $p, q \in \mathcal{P}$, if $q \in U_p = B_{\mathbb{R}^N}(p, r) \cap M$, where $r = \operatorname{rch}(M)/14$, then $\mathrm{pr}_{T_pM}(q) \in \mathrm{pr}_{T_pM}(\iota(\underline{\mathrm{St}}(p)))$ if and only if q is a vertex of $\underline{\mathrm{St}}(p)$.*

then the conclusions of Theorem 17 hold, and consequence (3) can be tightened to:

(3′) *For any* $x \in \boldsymbol{\sigma} \in \mathcal{A}$,

$$\delta_M(x) = |\check{x} - x| \leq 2\epsilon_0^2 \operatorname{rch}(M), \quad \textit{and} \quad \sin \angle(\boldsymbol{\sigma}, T_{\check{x}}) \leq \frac{3\epsilon_0}{t_0},$$

where $\check{x} = \operatorname{pr}_M(x)$.

Discussion

Theorem 16 can be applied to any map that could serve as a triangulation. However, we only know of two specific examples: the closest-point projection map featured in Theorem 17, and the barycentric coordinate map [11] mentioned in the introduction.

Conditions guaranteeing that the barycentric coordinate map is a triangulation were already established in [11], however the generic triangulation theorem [11, Proposition 16] on which the result is based had a flaw (discovered during the preparation of this current work), so that injectivity of the map is not guaranteed. This flaw has been repaired, using the vertex sanity criterion (Definition 14); this is described in [5, Appendix C], where the corrected statement of the theorem for the barycentric coordinate map can be found [5, Theorem 60].

Theorem 16 also leads to triangulation criteria for the barycentric coordinate map. However, the bound on the diameter of the simplices (the sampling radius) is proportional to the square of the thickness bound t_0 when Theorem 16 is employed, but it is only linear in t_0 when [11, Proposition 16] is used. This indicates that there is also room to improve the bound on ϵ_0 in Theorem 17(b) from quadratic to linear in t_0.

References

1 N. Amenta and M. Bern. Surface reconstruction by Voronoi filtering. *Discrete and Computational Geometry*, 22(4):481–504, 1999.

2 N. Amenta, S. Choi, T. K. Dey, and N. Leekha. A simple algorithm for homeomorphic surface reconstruction. *Int. J. Computational Geometry and Applications*, 12(2):125–141, 2002.

3 J.-D. Boissonnat, R. Dyer, and A. Ghosh. The stability of Delaunay triangulations. *International Journal of Computational Geometry & Applications*, 23(4-5):303–333, 2013. (arXiv:1304.2947).

4 J.-D. Boissonnat, R. Dyer, and A. Ghosh. Delaunay triangulation of manifolds. *Foundations of Computational Mathematics*, 2017. (arXiv:1311.0117).

5 J.-D. Boissonnat, R. Dyer, A. Ghosh, and M. Wintraecken. Local criteria for triangulation of manifolds. Technical Report 1803.07642, arXiv, 2017. URL: `http://arxiv.org/abs/1803.07642`.

6 J.-D. Boissonnat and A. Ghosh. Manifold reconstruction using tangential Delaunay complexes. *Discrete and Computational Geometry*, 51(1):221–267, 2014.

7 J.-D. Boissonnat, A. Lieutier, and M. Wintraecken. The reach, metric distortion, geodesic convexity and the variation of tangent spaces. Technical Report hal-01661227, Inria, Sophia-Antipolis, 2017. Accepted for SoCG 2018. URL: `https://hal.inria.fr/hal-01661227`.

8 J.-D. Boissonnat and S. Oudot. Provably good sampling and meshing of surfaces. *Graphical Models*, 67(5):405–451, 2005.

9 S. S. Cairns. On the triangulation of regular loci. *Annals of Mathematics. Second Series*, 35(3):579–587, 1934.

10 S.-W. Cheng, T. K. Dey, and E. A. Ramos. Manifold reconstruction from point samples. In *SODA*, pages 1018–1027, 2005.

11 R. Dyer, G. Vegter, and M. Wintraecken. Riemannian simplices and triangulations. *Geometriae Dedicata*, 179:91–138, 2015.

12 R. Dyer, H. Zhang, and T. Möller. Surface sampling and the intrinsic Voronoi diagram. *Computer Graphics Forum (Special Issue of Symp. Geometry Processing)*, 27(5):1393–1402, 2008.

13 H. Edelsbrunner and N. R. Shah. Triangulating topological spaces. *Int. J. Comput. Geometry Appl.*, 7(4):365–378, 1997.

14 J. R. Munkres. *Elementary differential topology*. Princton University press, second edition, 1968.

15 J. H. C. Whitehead. On C^1-complexes. *Ann. of Math*, 41(4):809–824, 1940.

16 H. Whitney. *Geometric Integration Theory*. Princeton University Press, 1957.

The Reach, Metric Distortion, Geodesic Convexity and the Variation of Tangent Spaces

Jean-Daniel Boissonnat
Université Côte d'Azur, INRIA
Sophia-Antipolis, France
jean-daniel.boissonnat@inria.fr

André Lieutier
Dassault Systemes
Aix-en-Provence, France
andre.lieutier@3ds.com

Mathijs Wintraecken
Université Côte d'Azur, INRIA
Sophia-Antipolis, France
m.h.m.j.wintraecken@gmail.com

Abstract

In this paper we discuss three results. The first two concern general sets of positive reach: We first characterize the reach by means of a bound on the metric distortion between the distance in the ambient Euclidean space and the set of positive reach. Secondly, we prove that the intersection of a ball with radius less than the reach with the set is geodesically convex, meaning that the shortest path between any two points in the intersection lies itself in the intersection. For our third result we focus on manifolds with positive reach and give a bound on the angle between tangent spaces at two different points in terms of the distance between the points and the reach.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Reach, Metric distortion, Manifolds, Convexity

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.10

Related Version A full version of this paper is available at `hal.inria.fr/hal-01661227`

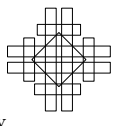
Funding This work has been partially funded by the European Research Council under the European Union's ERC Grant Agreement number 339025 GUDHI (Algorithmic Foundations of Geometric Understanding in Higher Dimensions).

Acknowledgements We are greatly indebted to Boris Thibert for suggestions and discussion. We also thank all members of the Datashape team (previously known as Geometrica), in particular Ramsay Dyer, Siddharth Pritam, and Mael Rouxel-Labbé for discussion. We would also like to thank the organizers of the workshop on 'Algorithms for the Medial Axis' at SoCG 2017 for allowing us to present some of the results of this paper.

1 Introduction

Metric distortion quantifies the maximum ratio between geodesic and Euclidean distances for pairs of points in a set $\mathcal{S}$. The reach of $\mathcal{S}$, defined by H. Federer [15], is the infimum of distances between points in $\mathcal{S}$ and points in its medial axis. Both reach and metric distortion are central concepts in manifold (re-)construction and have been used to characterize the

34th International Symposium on Computational Geometry (SoCG 2018).
Editors: Bettina Speckmann and Csaba D. Tóth; Article No. 10; pp. 10:1–10:14



Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

size of topological features. Amenta and Bern [1] introduced a local version of the reach in order to give conditions for homeomorphic surface reconstruction and this criterion has been used in many works aiming at topologically faithful reconstruction. See the seminal paper of Niyogi, Smale and Weinberger [19] and Dey's book [12] for more context and references. A direct relation between the reach and the size of topological features is simply illustrated by the fact that the intersection of a set with reach $r > 0$ with a ball of radius less than r has reach at least r and is contractible [3]. In a certain way, metric distortion also characterizes the size of topological features. This is illustrated by the fact that a compact subset of $\mathbb{R}^n$ with metric distortion less than $\pi/2$ is simply connected (Section 1.14 in [16] , see also appendix A by P. Pansu where sets with a given metric distortion are called *quasi convex sets*).

In the first part of this paper, we provide tight bounds on metric distortion for sets of positive reach and, in a second part, we consider submanifolds of $\mathbb{R}^d$ and bound the angle between tangent spaces at different points. Whenever we mention manifolds we shall tacitly assume that it is embedded in Euclidean space. Previous versions of the metric distortion result, restricted to the manifold setting can be found in [19]. A significant amount of attention has gone to tangent space variation, see [4, 6, 8, 11, 12, 13, 19] to name but a few.

Our paper improves on these bounds, extends the results beyond the case of smooth manifolds and offers new insights and results. These results have immediate algorithmic consequences by, on one hand, improving the sampling conditions under which known reconstruction algorithms are valid and, on the other hand, allowing us to extend the algorithms to the class of manifolds of positive reach, which is much larger than the usually considered class of C^2 manifolds. Indeed, the metric distortion and tangent variation bounds for $C^{1,1}$ manifolds presented in this paper in fact suffice to extend the triangulation result of C^2 manifolds embedded in Euclidean space given in [7] to arbitrary manifolds with positive reach, albeit with slightly worse constants.

Overview of results. For metric distortion, we extend and tighten the previously known results so much that our metric distortion result can be regarded as a completely new characterization of sets of positive reach. In particular, the standard manifold and smoothness assumptions are no longer necessary. Based on our new characterization of the reach by metric distortion, we can prove that the intersection of a set of positive reach with a ball with radius less than the reach is geodesically convex. This result is a far reaching extension of a result of [9] that has attracted significant attention, stating that, for smooth surfaces, the intersection is a pseudo-ball. Bounding the metric distortion may be a practical way to estimate the reach of a set in high ambient dimension.

To study tangent variation along manifolds, we will consider two different settings, namely the C^2 setting, for which the bounds are tight, and the $C^{1,1}$ setting, where we achieve slightly weaker bounds.

The exposition for C^2 manifolds is based on differential geometry and is a consequence of combining the work of Niyogi, Smale, and Weinberger [19], and the two dimensional analysis of Attali, Edelsbrunner, and Mileyko [2] with some observations concerning the reach. We would like to stress that some effort went into simplifying the exposition, in particular the part of [19] concerning the second fundamental form.

The second class of manifolds we consider consists of closed $C^{1,1}$ manifolds $\mathcal{M}$ embedded in $\mathbb{R}^d$. We restrict ourselves to $C^{1,1}$ manifolds because it is known that closed manifolds have positive reach if and only if they are $C^{1,1}$, see Federer [15, Remarks 4.20 and 4.21] and Scholtes [20] for a history of this result. Here we do not rely on differential geometry apart

from simple concepts such as the tangent space. In fact most proofs can be understood in terms of simple Euclidean geometry. Moreover our proofs are very pictorial. Although the bounds we attain are slightly weaker than the ones we attain using differential geometry, we should note that we have sometimes simplified the exposition at the cost of weakening the bound.

We also prove that the intersection of a $C^{1,1}$ manifold with a ball of radius less than the reach of the manifold is a topological ball. This is a generalization of previous results. A sketch of a proof of the result in the C^2 case has been given by Boissonnat and Cazals [5]. Our result also extends a related result of Attali and Lieutier [3]. It is furthermore related to the convexity result, but certainly not the same. This is because spaces can be geodesically convex without being topological disks, think for example of the equator of the sphere.

2 Metric distortion and convexity

For a closed set $\mathcal{S} \subset \mathbb{R}^d$, $d_{\mathcal{S}}$ denotes the geodesic distance in $\mathcal{S}$, i.e. $d_{\mathcal{S}}(a,b)$ is the infimum of lengths of paths in $\mathcal{S}$ between a and b. If there is at least one path between a and b with finite length, then it is known that a minimizing geodesic, i.e. a path with minimal length connecting a to b exists (see the second paragraph of part III, section 1: "Die Existenz geodätischer Bogen in metrischen Räumen" in [17]).

The next theorem can be read as an alternate definition of the reach, based on metric distortion. Observe that for fixed $|a-b|$, the function $r \mapsto 2r \arcsin \frac{|a-b|}{2r}$ is decreasing.

▶ **Theorem 1.** *If $\mathcal{S} \subset \mathbb{R}^d$ is a closed set, then*

$$\operatorname{rch} \mathcal{S} = \sup \left\{ r > 0,\ \forall a,b \in \mathcal{S},\ |a-b| < 2r \Rightarrow d_{\mathcal{S}}(a,b) \leq 2r \arcsin \frac{|a-b|}{2r} \right\},$$

where the sup *over the empty set is* 0.

Proof. Lemma 5 states that if $r' < \operatorname{rch} \mathcal{S}$ then

$$\forall a,b \in \mathcal{S},\ |a-b| < 2r' \Rightarrow d_{\mathcal{S}}(a,b) \leq 2r' \arcsin \frac{|a-b|}{2r'}.$$

This gives us

$$\sup \left\{ r > 0,\ \forall a,b \in \mathcal{S},\ |a-b| < 2r \Rightarrow d_{\mathcal{S}}(a,b) \leq 2r \arcsin \frac{|a-b|}{2r} \right\} \geq \operatorname{rch} \mathcal{S}.$$

If $\operatorname{rch} \mathcal{S} = \infty$, i.e. if $\mathcal{S}$ is convex, the theorem holds trivially. We assume now that the medial axis is non empty, i.e. $\operatorname{rch} \mathcal{S} < \infty$. Then by definition of the reach, if $r' > \operatorname{rch} \mathcal{S}$, there exists $x \in \mathbb{R}^d$ in the medial axis of $\mathcal{S}$ and $a,b \in \mathcal{S}, a \neq b$ such that $r' > r_x = d(x,\mathcal{S}) = d(x,a) = d(x,b)$. If for at least one of such pairs $\{a,b\}$ one has $d_{\mathcal{S}}(a,b) = \infty$ then $|a-b| \leq 2r_x < 2r'$ and:

$$\sup \left\{ r > 0,\ \forall a,b \in \mathcal{S},\ |a-b| < 2r \Rightarrow d_{\mathcal{S}}(a,b) \leq 2r \arcsin \frac{|a-b|}{2r} \right\} < r'$$

If not, consider a path γ in $\mathcal{S}$ between a and b: $\gamma(0) = a, \gamma(1) = b$. Because $\gamma([0,1])$ lies outside the open ball $B(x,r_x)^\circ$, its projection on the closed ball $B(x,r_x)$ cannot increase lengths. It follows that, for any $r \geq r'$:

$$d_{\mathcal{S}}(a,b) \geq 2r_x \arcsin \frac{|a-b|}{2r_x} > 2r \arcsin \frac{|a-b|}{2r}$$

which gives, for any $r' > \operatorname{rch}\mathcal{S}$,

$$\exists a, b \in \mathcal{S}, \forall r \geq r' \quad |a-b| < 2r \quad \text{and} \quad d_{\mathcal{S}}(a,b) > 2r \arcsin \frac{|a-b|}{2r},$$

and therefore

$$\sup \left\{ r > 0,\, \forall a, b \in \mathcal{S},\, |a-b| < 2r \Rightarrow d_{\mathcal{S}}(a,b) \leq 2r \arcsin \frac{|a-b|}{2r} \right\} \leq r'. \qquad \blacktriangleleft$$

▶ **Corollary 2.** *Let $\mathcal{S} \subset \mathbb{R}^d$ be a closed set with positive reach $r = \operatorname{rch}\mathcal{S} > 0$. Then, for any $r' < \operatorname{rch}\mathcal{S}$ and any $x \in \mathbb{R}^d$, if $B(x,r')$ is the closed ball centered at x with radius r', then $\mathcal{S} \cap B(x,r')$ is geodesically convex in $\mathcal{S}$.*

Proof. First it follows from the theorem that if $a, b \in \mathcal{S} \cap B(x,r')$, then $d_{\mathcal{S}}(a,b) < \infty$ which means that there exists a path of finite length in $\mathcal{S}$ between a and b. From [17] there is at least one minimizing geodesic in $\mathcal{S}$ between a and b.

For a contradiction assume that such a geodesic γ goes outside $B(x,r')$. In other words there is at least one non empty open interval (t_1,t_2) such that $\gamma(t_1), \gamma(t_2) \in \partial B(x,r')$ and $\gamma((t_1,t_2)) \cap B(x,r') = \emptyset$. But then, since the projection on the ball $B(x,r')$ reduces lengths, one has:

$$d_{\mathcal{S}}(\gamma(t_1), \gamma(t_2)) > 2r' \arcsin \frac{|\gamma(t_1) - \gamma(t_2)|}{2r'},$$

a contradiction with the theorem. $\blacktriangleleft$

2.1 Projection of the middle point

For a closed set $\mathcal{S} \subset \mathbb{R}^d$ with positive reach $r = \operatorname{rch}\mathcal{S} > 0$ and a point $m \in \mathbb{R}^d$ with $d(m,\mathcal{S}) < r$, $\pi_{\mathcal{S}}(m)$ denotes the projection of m on $\mathcal{S}$ as depicted on Figure 1 on the left.

▶ **Lemma 3.** *Let $\mathcal{S} \subset \mathbb{R}^d$ be a closed set with reach $r = \operatorname{rch}\mathcal{S} > 0$. For $a, b \in \mathcal{S}$ such that $\delta = \frac{|a-b|}{2} < r$ and $m = \frac{a+b}{2}$ one has $|\pi_{\mathcal{S}}(m) - m| \leq \rho$, with $\rho = r - \sqrt{r^2 - \delta^2}$.*

The disk of center m and radius ρ appears in green in Figure 1 left and right.

Proof. We shall now use a consequence of Theorem 4.8 of [15]. In the following section we shall discuss this result for the manifold setting, where it generalizes the tubular neighbourhood results for C^2 manifolds from differential geometry and differential topology. For the moment we restrict ourselves to the following: If $\pi_{\mathcal{S}}(m) \neq m$ claim (12) in Theorem 4.8 of [15] gives us:

$$\forall \lambda \in [0,r),\, \pi_{\mathcal{S}}\left(\pi_{\mathcal{S}}(m) + \lambda \frac{m - \pi_{\mathcal{S}}(m)}{|m - \pi_{\mathcal{S}}(m)|}\right) = \pi_{\mathcal{S}}(m),$$

which means that for $\lambda \in [0,r)$:

$$y(\lambda) = \pi_{\mathcal{S}}(m) + \lambda \frac{m - \pi_{\mathcal{S}}(m)}{|m - \pi_{\mathcal{S}}(m)|}$$

is closer to $\pi_{\mathcal{S}}(m)$ than both to a and to b (see Figure 1).

Without loss of generality we assume that $|a - \pi_{\mathcal{S}}(m)| \geq |b - \pi_{\mathcal{S}}(m)|$. We denote $\mu = |\pi_{\mathcal{S}}(m) - m|$ and want to prove that $\mu \leq \rho$.

In the plane spanned by $a, b, \pi_{\mathcal{S}}(m)$ we consider the following frame $(m, \frac{a-m}{|a-m|}, \tau)$, where m denotes the origin, τ is a unit vector orthogonal to $a-m$ and such that $\langle \tau, \pi_{\mathcal{S}}(m) - m \rangle \leq 0$.

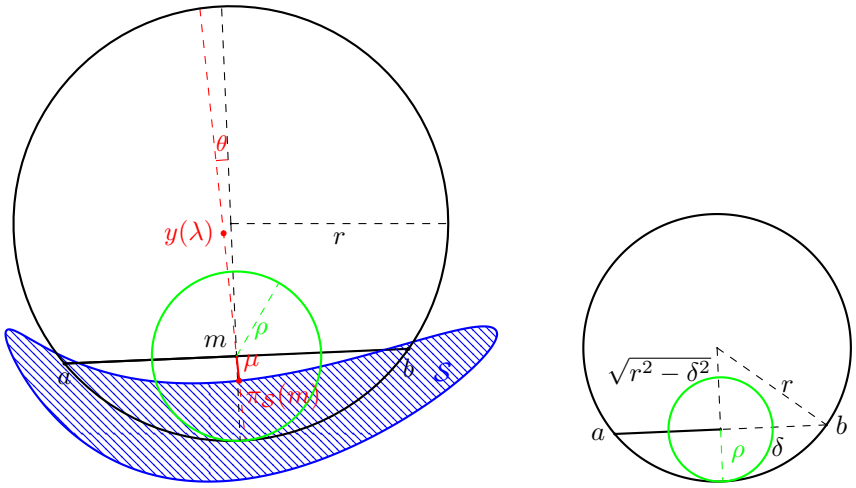


Figure 1 On the left the projection $\pi_{\mathcal{S}}(m)$ is contained in the disk of center m and radius ρ. The notation used in the proof of Lemma 3 is also added. From the right figure it is easy to deduce that $\rho = r - \sqrt{r^2 - \delta^2}$.

For some $\theta \in [0, \pi/2]$, the coordinates of $\pi_{\mathcal{S}}(m)$ in the frame are $(-\mu \sin\theta, -\mu\cos\theta)$. The coordinate of a are $(\delta, 0)$ and the coordinates of $y(\lambda)$ are, as shown in Figure 1, $((\lambda-\mu)\sin\theta, (\lambda-\mu)\cos\theta)$. Since $y(\lambda)$ is closer to $\pi_{\mathcal{S}}(m)$ than to a, one has

$$\forall \lambda \in [0, r), \quad (\delta - (\lambda-\mu)\sin\theta)^2 + (\lambda-\mu)^2\cos^2\theta > \lambda^2.$$

This is a degree 2 inequality in μ. One gets, for any $\lambda \in [0, r)$, if $\Delta \geq 0$,

$$\mu \notin \left[(\lambda - \delta\sin\theta) - \sqrt{\Delta},\ (\lambda - \delta\sin\theta) + \sqrt{\Delta}\right],$$

with $\Delta = (\lambda - \delta\sin\theta)^2 - (\delta^2 - 2\delta\lambda\sin\theta) = \lambda^2 - \delta^2 + (\delta\sin\theta)^2$. For $\lambda \geq \delta$ one has $\Delta \geq \lambda^2 - \delta^2$. Therefore: $(\lambda - \delta\sin\theta) - \sqrt{\Delta} \leq \lambda - \sqrt{\lambda^2 - \delta^2}$ and since $\lambda \mapsto \lambda - \sqrt{\lambda^2 - \delta^2}$ is continuous, one has:

$$\inf_{\lambda < r} \left\{(\lambda - \delta\sin\theta) - \sqrt{\Delta}\right\} \leq r - \sqrt{r^2 - \delta^2} = \rho,$$

also, when $\lambda \geq \delta$ one has $\sqrt{\Delta} \geq \delta\sin\theta$ and $(\lambda - \delta\sin\theta) + \sqrt{\Delta} \geq \delta$. Since $\mu \leq d(m, a) = \delta$, one finds that $\mu \leq \rho$. ◀

The following simple geometric Lemma is used in the next section.

▶ **Lemma 4.** *Consider a circle $\tilde{C}$ of radius r and two points $a, b \in \tilde{C}$ with $|a-b|/2 = \delta < r$. Define the middle point $m = \frac{a+b}{2}$ and consider a point p such that $|p-m| \leq \rho = r - \sqrt{r^2 - \delta^2}$. Denote $\tilde{C}_{a,b}$ the shortest of the arcs of the circle in $\tilde{C}$ bounded by a and b. Define $\tilde{p} \in \tilde{C}_{a,b}$ as the unique point in $\tilde{C}_{a,b}$ such that $\frac{|a-\tilde{p}|}{|b-\tilde{p}|} = \frac{|a-p|}{|b-p|}$, then we have $|a-p| \leq |a-\tilde{p}|$ and $|b-p| \leq |b-\tilde{p}|$.*

The proof of this lemma is fairly straightforward and can be found in the appendix of [10].

2.2 Upper bound on geodesic length

In this section we establish an upper bound on geodesic lengths through the iterative construction of a sequence of paths.

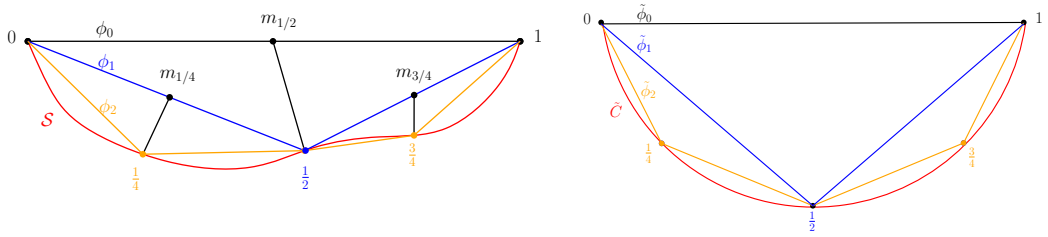


Figure 2 Left: ϕ_0, ϕ_1, ϕ_2 , Right: $\tilde{\phi}_0, \tilde{\phi}_1, \tilde{\phi}_2$.

▶ **Lemma 5.** *Let $\mathcal{S} \subset \mathbb{R}^d$ be a closed set with reach $r = \operatorname{rch} \mathcal{S} > 0$. For any $a, b \in \mathcal{S}$ such that $|a-b| < 2r$ one has $d_{\mathcal{S}}(a,b) \leq 2r \arcsin \frac{|a-b|}{2r}$.*

Proof. We build two sequences of PL-functions (see Figure 2). For $i \in \mathbb{N}$, $\phi_i : [0,1] \to \mathbb{R}^d$ and $\tilde{\phi}_i : [0,1] \to \mathbb{R}^2$ are defined as follows.

First we define $\phi_0(t) = a + t(b-a)$. Denote $m = \frac{a+b}{2}$ the middle point of $[a,b]$. Since $d(m, \mathcal{S}) \leq d(m,a) = \delta < r$, the point $p = \pi_{\mathcal{S}}(m)$ is well defined. Secondly, we define

$$\phi_1(t) = \begin{cases} a + 2t(p-a) & \text{if} \quad t \leq 1/2 \\ p + (2t-1)(b-p) & \text{if} \quad t \geq 1/2. \end{cases}$$

as depicted in Figure 2 on the left.

From Lemma 3, one has $|p-m| \leq \rho = r - \sqrt{r^2 - \delta^2} < r$ and thus

$$\min\left(|a-p|, |b-p|\right) \geq \delta - \rho > 0 \qquad \max\left(|a-p|, |b-p|\right) \leq \delta + \rho$$

We also fix a circle $\tilde{C}$ in $\mathbb{R}^2$ with radius r and we consider $\tilde{a}, \tilde{b} \in \mathbb{R}^2$ such that $\tilde{a}, \tilde{b} \in \tilde{C}$ and $|\tilde{a} - \tilde{b}| = |a-b|$ and we define $\tilde{\phi}_0(t) = \tilde{a} + t(\tilde{b} - \tilde{a})$. Denote by $\tilde{C}_{\tilde{a},\tilde{b}}$ the shortest of the two arcs of $\tilde{C}$ bounded by $\tilde{a}, \tilde{b}$ and $\tilde{p}$ as constructed in Lemma 4 i.e. $\tilde{p} \in \tilde{C}_{\tilde{a},\tilde{b}}$ such that $\frac{|\tilde{p}-\tilde{a}|}{|\tilde{p}-\tilde{b}|} = \frac{|p-a|}{|p-b|}$, as shown in Figure 2 on the right, and define

$$\tilde{\phi}_1(t) = \begin{cases} \tilde{a} + 2t(\tilde{p} - \tilde{a}) & \text{if} \quad t \leq 1/2 \\ \tilde{p} + (2t-1)(\tilde{b} - \tilde{p}) & \text{if} \quad t \geq 1/2. \end{cases}$$

Applying Lemma 4 we get $|a-p| \leq |\tilde{a} - \tilde{p}|$, $|b-p| \leq |\tilde{b} - \tilde{p}|$, and

$$\operatorname{length}(\phi_1) = |a-p| + |b-p| \leq |\tilde{a} - \tilde{p}| + |\tilde{b} - \tilde{p}| = \operatorname{length}(\tilde{\phi}_1).$$

For $i \geq 2$, ϕ_i and $\tilde{\phi}_i$ are PL functions with 2^i intervals. For $k \in \mathbb{N}$, $0 \leq k \leq 2^i$, $\phi_i(k/2^i) \in \mathcal{S}$, $\tilde{\phi}_i(k/2^i) \in \tilde{C}_{\tilde{a},\tilde{b}}$ are defined by applying to each of the 2^{i-1} segments of $\phi_{i-1}([0,1])$ and $\tilde{\phi}_{i-1}([0,1])$ the same subdivision process used when defining ϕ_1 and $\tilde{\phi}_1$.

If k is even we set $\phi_i(k/2^i) = \phi_{i-1}(k/2^i)$ and $\tilde{\phi}_i(k/2^i) = \tilde{\phi}_{i-1}(k/2^i)$.

If k is odd define:

$$m_{k/2^i} = \frac{\phi_i((k-1)/2^i) + \phi_i((k+1)/2^i)}{2} \quad \text{and} \quad \phi_i(k/2^i) = \pi_{\mathcal{S}}\left(m_{k/2^i}\right).$$

Note that $m_{1/2}$ corresponds to m defined above.

Let $\tilde{\phi}_i(k/2^i) \in \tilde{C}_{\tilde{\phi}_{i-1}((k-1)/2^i),\tilde{\phi}_{i-1}((k+1)/2^i)} \subset \tilde{C}_{\tilde{a},\tilde{b}}$ be such that:

$$\frac{|\tilde{\phi}_i(k/2^i) - \tilde{\phi}_{i-1}((k-1)/2^i)|}{|\tilde{\phi}_i(k/2^i) - \tilde{\phi}_{i-1}((k+1)/2^i)|} = \frac{|\phi_i(k/2^i) - \phi_{i-1}((k-1)/2^i)|}{|\phi_i(k/2^i) - \phi_{i-1}((k+1)/2^i)|}.$$

Figure 2 left shows the curves ϕ_1 and ϕ_2 in blue and yellow respectively.

Applying Lemma 4, since by induction,

$$\left|\phi_{i-1}((k+1)/2^{i-1}) - \phi_{i-1}(k/2^{i-1})\right| \leq \left|\tilde{\phi}_{i-1}((k+1)/2^{i-1}) - \tilde{\phi}_{i-1}(k/2^{i-1})\right|$$

we get that for $i \in \mathbb{N}$ and $p = 0, \ldots, 2^i - 1$:

$$|\phi_i((k+1)/2^i) - \phi_i(k/2^i)| \leq |\tilde{\phi}_i((k+1)/2^i) - \tilde{\phi}_i(k/2^i)|,$$

and therefore:

$$\begin{aligned} \operatorname{length}(\phi_i) &= \sum_{k=0}^{2^i-1} |\phi_i((k+1)/2^i) - \phi_i(k/2^i)| \\ &\leq \sum_{k=0}^{2^i-1} |\tilde{\phi}_i((k+1)/2^i) - \tilde{\phi}_i(k/2^i)| \\ &= \operatorname{length}(\tilde{\phi}_i) \leq \operatorname{length}(\tilde{C}_{\tilde{a},\tilde{b}}) = 2r \arcsin \frac{|a-b|}{2r}. \end{aligned} \tag{1}$$

We study now the behavior of the sequence $\phi_i, i \in \mathbb{N}$. Define $\delta_0 = \delta$ and $\rho_0 = \rho$. Further define δ_i as

$$\delta_i = \frac{1}{2} \max_{0 \leq k \leq 2^i - 1} |\phi_i((k+1)/2^i) - \phi_i(k/2^i)|.$$

i.e. half the max of lengths of all segments of $\phi_i([0,1])$ and $\rho_i = r - \sqrt{r^2 - \delta_i^2}$. We make the following assertion:

▶ **Claim 6.**

$$\lim_{i \to \infty} \delta_i = 0. \tag{2}$$

The proof of this claim is given in the appendix of [10].

Since for any $i \geq 0$ and $t \in [0,1]$, $d(\phi(t), \mathcal{S}) \leq \delta_i$ and $\delta_i < \operatorname{rch} \mathcal{S}$ the curves $\pi_{\mathcal{S}} \circ \phi_i$, (projections of ϕ_i on $\mathcal{S}$) are well defined, with $\pi_{\mathcal{S}} \circ \phi_i : [0,1] \to \mathcal{S}$, $\pi_{\mathcal{S}} \circ \phi_i(0) = a$ and $\pi_{\mathcal{S}} \circ \phi_i(1) = b$.

Claim (8) in Theorem 4.8 of [15] states that for $\mu < r = \operatorname{rch}\mathcal{S}$ the restriction of $\pi_{\mathcal{S}}$ to the μ-tubular neighbourhood $\mathcal{S}^\mu$ is $\frac{\operatorname{rch}\mathcal{S}}{\operatorname{rch}\mathcal{S} - \mu}$-Lipschitz. This together with (1) above gives us an upper bound on the lengths of curves $\pi_{\mathcal{S}} \circ \phi_i$:

$$\operatorname{length}(\pi_{\mathcal{S}} \circ \phi_i) \leq \frac{\operatorname{rch} \mathcal{S}}{\operatorname{rch} \mathcal{S} - \delta_i} \operatorname{length}(\phi_i) \leq \frac{\operatorname{rch} \mathcal{S}}{\operatorname{rch} \mathcal{S} - \delta_i} 2r \arcsin \frac{|a-b|}{2r}$$

This together with (2) yields $d_{\mathcal{S}}(a,b) \leq 2r \arcsin \frac{|a-b|}{2r}$. ◀

3 Variation of tangent spaces

In this section we shall first discuss the bound on the variation of tangent spaces in the C^2 setting, and then generalize to the $C^{1,1}$ setting. For this generalization we need a topological result, which will be presented in Section 3.2.

3.1 Bounds for C^2 submanifolds

We shall be using the following result, Theorem 4.8(12) of [15]:

▶ **Theorem 7** (Federer's tubular neighbourhoods). *Let $B_{N_p\mathcal{M}}(r)$, be the ball of radius r centred at p in the normal space $N_p\mathcal{M} \subset \mathbb{R}^d$ of a $C^{1,1}$ manifold $\mathcal{M}$ with reach $\operatorname{rch}(\mathcal{M})$, where $r < \operatorname{rch}(\mathcal{M})$. For every point $x \in B_{N_p\mathcal{M}}(r)$, $\pi_{\mathcal{M}}(x) = p$.*

The fact that such a tubular neighbourhood exists is non-trivial, even for a neighbourhood of size ϵ. From Theorem 7 we immediately see that:

▶ **Corollary 8.** *Let $\mathcal{M}$ be a submanifold of $\mathbb{R}^d$ and $p \in \mathcal{M}$. Any open ball $B(c, r)$ that is tangent to $\mathcal{M}$ at p and whose radius r satisfies $r \leq \operatorname{rch}(\mathcal{M})$ does not intersect $\mathcal{M}$.*

Proof. Let $r < \operatorname{rch}(\mathcal{M})$. Suppose that the intersection of $\mathcal{M}$ and the open ball is not empty, then the $\pi_{\mathcal{M}}(c) \neq p$ contradicting Federer's tubular neighbourhood theorem. The result for $r = \operatorname{rch}(\mathcal{M})$ now follows by taking the limit. ◀

Here we prove the main result for C^2 manifolds. Our exposition is the result of straightforwardly combining the work of Niyogi, Smale, and Weinberger [19], and the two dimensional analysis of Attali, Edelsbrunner, and Mileyko [2] with some observations concerning the reach.

We start with the following simple observation:

▶ **Lemma 9.** *Let $\gamma(t)$ be a geodesic parametrized according to arc length on $\mathcal{M} \subset \mathbb{R}^d$, then $|\ddot{\gamma}| \leq 1/\operatorname{rch}(\mathcal{M})$, where we use Newton's notation, that is we write $\ddot{\gamma}$ for the second derivative of γ with respect to t.*

Proof. Because $\gamma(t)$ is a geodesic, $\ddot{\gamma}(t)$ is normal to $\mathcal{M}$ at $\gamma(t)$. Now consider the sphere of radius $\operatorname{rch}(\mathcal{M})$ tangent to $\mathcal{M}$ at $\gamma(t)$, whose centre lies on the line $\{\gamma(t) + \lambda\ddot{\gamma} \mid \lambda \in \mathbb{R}\}$. If now $|\ddot{\gamma}|$ were larger than $1/\operatorname{rch}(\mathcal{M})$, the geodesic γ would enter the tangent sphere, which would contradict Corollary 8. ◀

Note that $|\ddot{\gamma}|$ is the normal curvature, because γ is a geodesic. Using the terminology of [19, Section 6], Lemma 9 can also be formulated as follows: $1/\operatorname{rch}(\mathcal{M})$ bounds the principal curvatures in the normal direction ν, for any unit normal vector $\nu \in N_p\mathcal{M}$. In particular, $1/\operatorname{rch}(\mathcal{M})$ also bounds the principal curvatures if $\mathcal{M}$ has codimension 1.

We now have the following, which is a straightforward extension of an observation in [2] to general dimension:

▶ **Lemma 10.** *Let $\gamma(t)$ be a geodesic parametrized according to arc length, with $t \in [0, \ell]$ on $\mathcal{M} \subset \mathbb{R}^d$, then:*

$$\angle\dot{\gamma}(0)\dot{\gamma}(\ell) \leq \frac{d_{\mathcal{M}}(\gamma(0), \gamma(\ell))}{\operatorname{rch}(\mathcal{M})}.$$

Proof. Because γ is parametrized according to arc length $|\dot{\gamma}| = 1$ and $\dot{\gamma}(t)$ can be seen as a curve on the sphere $\mathbb{S}^{d-1}$. Moreover $\ddot{\gamma}$ can be seen as tangent to this sphere. The angle between two tangent vectors $\dot{\gamma}(0)$ and $\dot{\gamma}(\ell)$ equals the geodesic distance on the sphere. The geodesic distance between any two points is smaller or equal to the length of any curve connecting these points, and $\{\dot{\gamma}(t) \mid t \in [0, \ell]\}$ is such a curve. We therefore have

$$\angle\dot{\gamma}(0)\dot{\gamma}(\ell) \leq \int_0^\ell \left|\frac{d}{dt}\dot{\gamma}\right| \mathrm{d}t = \int_0^\ell |\ddot{\gamma}|\mathrm{d}t \leq \frac{\ell}{\operatorname{rch}(\mathcal{M})} \leq \frac{d_{\mathcal{M}}(\gamma(0), \gamma(\ell))}{\operatorname{rch}(\mathcal{M})}, \tag{3}$$

where we used Lemma 9. ◀

We can now turn our attention to the variation of tangent spaces. Here we mainly follow Niyogi, Smale, and Weinberger [19], but use one useful observation of [2]. We shall be using the second fundamental form, which we assume the reader to be familiar with. We refer to [14] as a standard reference.

The second fundamental form $\mathrm{I\!I}_p(u,v)$ has the geometric interpretation of the normal part of the covariant derivative, where we assume now that u, v are vector fields. In particular $\mathrm{I\!I}(u,v) = \bar{\nabla}_u v - \nabla_u v$, where $\bar{\nabla}$ is the connection in the ambient space, in this case Euclidean space, and ∇ the connection with respect to the induced metric on the manifold $\mathcal{M}$. The second fundamental form $\mathrm{I\!I}_p : T_p\mathcal{M} \times T_p\mathcal{M} \to N_p\mathcal{M}$ is a symmetric bi-linear form, see for example Section 6.2 of [14] for a proof. This means that we only need to consider vectors in the tangent space and not vector fields, when we consider $\mathrm{I\!I}_p(u,v)$.

We can now restrict our attention to u, v lying on the unit sphere $\mathbb{S}^{n-1}_{T_p\mathcal{M}}$ in the tangent space and ask for which of these vectors $|\mathrm{I\!I}_p(u,v)|$ is maximized. Let us assume that the $\mathrm{I\!I}_p(u,v)$ for which the maximum[1] is attained lies in the direction of $\eta \in N_p\mathcal{M}$ where η is assumed to have unit length.

We can now identify $\langle \mathrm{I\!I}_p(\cdot,\cdot), \eta\rangle$, with a symmetric matrix. Because of this $\langle \mathrm{I\!I}_p(u,v), \eta\rangle$, with $u, v \in \mathbb{S}^{n-1}_{T_p\mathcal{M}}$, attains its maximum for u, v both lying in the direction of the unit eigenvector w of $\langle \mathrm{I\!I}_p(\cdot,\cdot), \eta\rangle$ with the largest[2] eigenvalue. In other words the maximum is assumed for $u = v = w$. Let us now consider a geodesic γ_w on $\mathcal{M}$ parametrized by arclength such that $\gamma_w(0) = p$ and $\dot{\gamma}_w(0) = w$. Now, because γ_w is a geodesic and the ambient space is Euclidean,

$$\mathrm{I\!I}_p(w,w) = \mathrm{I\!I}_p(\dot{\gamma}_w, \dot{\gamma}_w) = \bar{\nabla}_{\dot{\gamma}_w}\dot{\gamma}_w - \nabla_{\dot{\gamma}_w}\dot{\gamma}_w = \bar{\nabla}_{\dot{\gamma}_w}\dot{\gamma}_w - 0 = \ddot{\gamma}_w.$$

Due to Lemma 9 and by definition of the maximum, we now see that $|\mathrm{I\!I}_p(u,v)| \leq |\mathrm{I\!I}_p(w,w)| \leq 1/\mathrm{rch}\mathcal{M}$, for all u, v of length one.

Having discussed the second fundamental form, we can give the following lemma:

▶ **Lemma 11.** *Let $p, q \in \mathcal{M}$, then*

$$\angle(T_p\mathcal{M}, T_q\mathcal{M}) \leq \frac{d_{\mathcal{M}}(p,q)}{\mathrm{rch}(\mathcal{M})}.$$

Proof. Let γ be a geodesic connecting p and q, parametrized by arc length. We consider an arbitrary unit vector u and parallel transport this unit vector along γ, getting the unit vectors $u(t)$ in the tangent spaces $T_{\gamma(t)}\mathcal{M}$. The maximal angle between $u(0)$ and $u(\ell)$, for all u bounds the angle between $T_p\mathcal{M}$ and $T_q\mathcal{M}$. Now

$$\frac{du}{dt} = \bar{\nabla}_{\dot{\gamma}} u(t) = \mathrm{I\!I}_{\gamma(t)}(\dot{\gamma}, u(t)) + \nabla_{\dot{\gamma}} u(t) = \mathrm{I\!I}_{\gamma(t)}(\dot{\gamma}, u(t)) + 0,$$

where we used that $u(t)$ is parallel and thus by definition $\nabla_{\dot{\gamma}} u(t) = 0$. So using our discussion above $|\frac{du}{dt}| \leq 1/\mathrm{rch}(\mathcal{M})$. Now we note that, similarly to what we have seen in the proof of Lemma 10, $u(t)$ can be seen as a curve on the sphere and thus $\angle(u(0), u(\ell)) \leq \int_0^\ell |\frac{du}{dt}| \mathrm{d}t \leq \ell/\mathrm{rch}(\mathcal{M})$. ◀

This bound is tight as it is attained for a sphere.

Combining Theorems 1 and 11 we find that

[1] If there is more than one direction we simply pick one.

[2] We can assume positivity without loss of generality, and, again, if there is more than one direction, we pick one.

▶ **Corollary 12.**

$$\sin\left(\frac{\angle(T_p\mathcal{M}, T_q\mathcal{M})}{2}\right) \leq \frac{|p-q|}{2\mathrm{rch}(\mathcal{M})}.$$

The proof is almost immediate, but has been added to the appendix of [10] for completeness.

With the bound on the angles between the tangent spaces it is not difficult to prove that the projection map onto the tangent space is locally a diffeomorphism, as has been done in [19]. Although the results were given in terms of the (global) reach to simplify the exposition, the results can be easily formulated in terms of the local feature size.

3.2 A topological result

We shall now give a full proof of a statement by Boissonnat and Cazals [5, Proposition 12] in the more general $C^{1,1}$ setting:

▶ **Proposition 13.** *Let B be a closed ball that intersects a $C^{1,1}$ manifold $\mathcal{M}$. If B does not contain a point of the medial axis ($ax(\mathcal{M})$) of $\mathcal{M}$ then $B \cap \mathcal{M}$ is a topological ball.*

The proof uses some results from topology, namely variations of [18, Theorem 3.1 and Theorem 3.2]:

▶ **Lemma 14.** *Consider the distance function from c: $d_c : \mathbb{R}^d \to \mathbb{R}, d_c(x) = |x - c|$ restricted to $\mathcal{M}$. Let $a = d_c(x')$ and $b = r$ and suppose that the set $d_c^{-1}[a, b]$, consisting of all $p \in \mathcal{M}$ with $a \leq d_c(p) \leq b$, contains no critical points of d_c (that is, no point q of $\mathcal{M}$ where $B(c, q)$ is tangent to $\mathcal{M}$). Then $\mathcal{M}^a = \{x \in \mathcal{M}, d_c \leq a\} = \mathcal{M} \cap B(c, a)$ is homeomorphic (if d_c is $C^{1,1}$) to $\mathcal{M}^b = \{x \in \mathcal{M}, d_c \leq b\}$. Furthermore M^a is a deformation retract of M^b.*

▶ **Lemma 15.** *Let $d_c|_{\mathcal{M}}$ be the $C^{1,1}$ function on $\mathcal{M}$ defined, as in Lemma 14, as the restriction to $\mathcal{M}$ of $d_c : \mathbb{R}^d \to \mathbb{R}, d_c(x) = |x - c|$. Assume that y is a global isolated minimum of $d_c|_{\mathcal{M}}$ and let r_c be the second critical value of $d_c|_{\mathcal{M}}$. Then for all $0 < \eta < r_c - |c - y|$, $\mathcal{M}^{r_c - \eta}$ is a topological ball.*

The proofs of these lemmas can be found in the appendix of [10].

Proof of Proposition 13. Write r for the radius of B and c for its center. The result is trivial if c belongs to the medial axis of $\mathcal{M}$. Therefore assume that $c \notin \mathrm{axis}(\mathcal{M})$.

Let y be the (unique) point of $\mathcal{M}$ closest to c. We denote by B_y the closed ball centered at c with radius $|c - y|$ (see Figure 3). By Corollary 8, the interior of B_y does not intersect $\mathcal{M}$ and $B_y \cap \mathcal{M} = \{y\}$. This means that the conditions of Lemma 15 are satisfied and $B(c, r_c - \eta) \cap \mathcal{M}$ is a topological ball for all $0 < \eta < r_c - |c - y|$, where r_c is the second critical value of the distance function to c restricted to $\mathcal{M}$. In other words r_c is the radius for which the ball centred on c is tangent to $\mathcal{M}$ for the second time.

Let us now assume that there exists a point $z \neq y$ of $\mathcal{M}$ such that $r_c = |c - z| > |c - y|$ where the ball $B(c, r_c)$ is tangent to $\mathcal{M}$. We consider the set $\mathcal{B}_z$ of closed balls that are tangent to $\mathcal{M}$ at z and are centred on the line segment $[zc]$. The balls in $\mathcal{B}_z$ can be ordered according to their radius. Note that $B(c, r_c)$ is the ball of $\mathcal{B}_z$ centered at c. Since the interior of $B(c, r_c)$ contains y and therefore intersects $\mathcal{M}$, there must exist a largest ball $B_z \in \mathcal{B}_z$, whose interior does not intersect $\mathcal{M}$. The center of B_z belongs to both $\mathrm{ax}(\mathcal{M})$ and B since $B_z \subset B(c, r_c) \subset B$. ◀

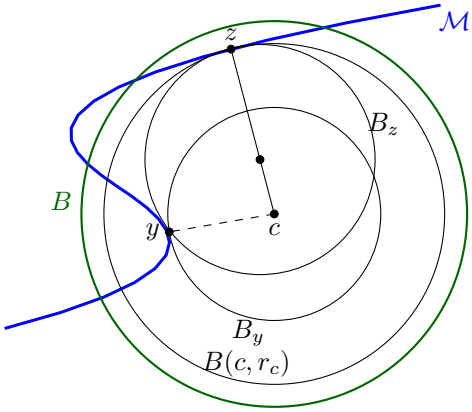


Figure 3 For the proof of Proposition 13.

3.3 Bounds for $C^{1,1}$ submanifolds

We shall now give an elementary exposition, in the sense that we do not rely on differential geometry, for the result of the previous section.

3.3.1 From manifold to tangent space and back

We start with the following lemma, which is due to Federer. It bounds the distance of a point $q \in \mathcal{M}$ to the tangent space of a point that is not too far away.

▶ **Lemma 16** (Distance to tangent space, Theorem 4.8(7) of [15]). *Let $p, q \in \mathcal{M} \subset \mathbb{R}^d$ such that $|p-q| < \operatorname{rch}(M)$. We have*

$$\sin \angle([pq], T_p\mathcal{M}) \leq \frac{|p-q|}{2\operatorname{rch}(\mathcal{M})}, \tag{4}$$

and

$$d_{\mathbb{E}}(q, T_p\mathcal{M}) \leq \frac{|p-q|^2}{2\operatorname{rch}(\mathcal{M})}. \tag{5}$$

We also have the converse statement of the distance bounds in Lemma 16. The following lemma is an improved version of Lemma B.2 in [6]. This result too can be traced back to Federer [15], in a slightly different guise. Before we give the lemma we first introduce the following notation. Let $C(T_p\mathcal{M}, r_1, r_2)$ denote the 'filled cylinder' given by all points that project orthogonally onto a ball of radius r_1 in $T_p\mathcal{M}$ and whose distance to this ball is less than r_2.

In the following lemma we prove for all points $v \in T_p\mathcal{M}$, such that $|v-p|$ is not too large, that a pre-image on $\mathcal{M}$, if it exists, under the projection to $T_p\mathcal{M}$ cannot be too far from $T_p\mathcal{M}$. The existence of such a point on $\mathcal{M}$ is proven below.

▶ **Lemma 17** (Distance to Manifold). *Suppose that $v \in T_p\mathcal{M}$ and $|v-p| < \operatorname{rch}(\mathcal{M})$. Let $q = \pi^{-1}_{(\mathcal{M}\to T_p\mathcal{M})}(v)$ be the inverse of the (restricted) projection $\pi_{T_p\mathcal{M}}$ from the intersection $\mathcal{M} \cap C(T_p\mathcal{M}, \operatorname{rch}(\mathcal{M}), \operatorname{rch}(\mathcal{M}))$ to $T_p\mathcal{M}$ of v, if it exists. Then*

$$|q-v| \leq \left(1 - \sqrt{1 - \left(\frac{|v-p|}{\operatorname{rch}(\mathcal{M})}\right)^2}\right) \operatorname{rch}(\mathcal{M}) \leq \frac{1}{2}\frac{|v-p|^2}{\operatorname{rch}(\mathcal{M})} + \frac{1}{2}\frac{|v-p|^4}{\operatorname{rch}(\mathcal{M})^3}.$$

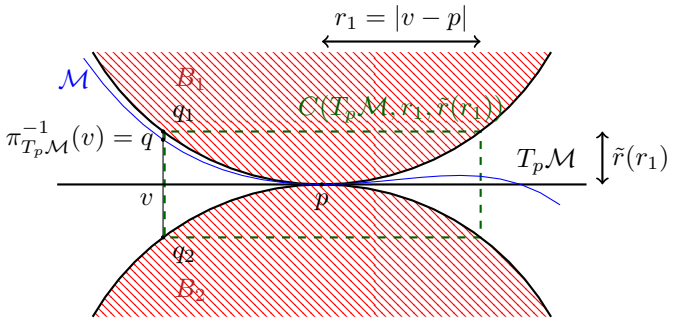


Figure 4 The set of all tangent balls to the tangent space of radius $\mathrm{rch}(\mathcal{M})$ bounds the region in which $\mathcal{M}$ can lie. Here we depict the 2 dimensional analogue.

▶ Remark 18. It follows immediately that $\mathcal{M} \cap C(T_p\mathcal{M}, r_1, \mathrm{rch}(\mathcal{M})) \subset C(T_p\mathcal{M}, r_1, \tilde{r}(r_1))$, with

$$\tilde{r}(r_1) = \left(1 - \sqrt{1 - \left(\frac{r_1}{\mathrm{rch}(\mathcal{M})}\right)^2}\right) \mathrm{rch}(\mathcal{M}). \tag{6}$$

This cylinder is indicated in green in Figure 4. Let $C_{\text{top/bottom}}(T_p\mathcal{M}, r_1, \tilde{r}(r_1))$ denote the subset of $C(T_p\mathcal{M}, r_1, \tilde{r}(r_1))$ that projects orthogonally onto the open ball of radius r_1 in $T_p\mathcal{M}$ and lies a distance $\tilde{r}(r_1)$ away. We also see that $\mathcal{M} \cap C_{\text{top/bottom}}(T_p\mathcal{M}, r_1, \tilde{r}(r_1)) = \emptyset$ and that $\mathcal{M} \cap C(T_p\mathcal{M}, r_1, \mathrm{rch}(\mathcal{M})) \cap N_p\mathcal{M} = \{p\}$. We write

$$C_{\text{side rim}}(T_p\mathcal{M}, r_1, \tilde{r}(r_1)) = \partial C(T_p\mathcal{M}, r_1, \tilde{r}(r_1)) \setminus C_{\text{top/bottom}}(T_p\mathcal{M}, r_1, \tilde{r}(r_1)).$$

3.3.2 The angle bound

This section revolves around the following observation: If r_1 roughly the distance between p and q, there is a significant part of $\mathcal{M}$ that is contained in the intersection $C(T_p\mathcal{M}, r_1, \tilde{r}) \cap C(T_q\mathcal{M}, r_1, \tilde{r})$. In particular any line segment, whose length is denoted by ℓ, connecting two points in $\mathcal{M} \cap C(T_p\mathcal{M}, r_1, \tilde{r}) \cap C(T_q\mathcal{M}, r_1, \tilde{r})$ is contained in both $C(T_p\mathcal{M}, r_1, \tilde{r})$ and $C(T_q\mathcal{M}, r_1, \tilde{r})$. If this line segment is long, the angle with both $T_p\mathcal{M}$ and $T_q\mathcal{M}$ is small. This bounds the angle between $T_p\mathcal{M}$ and $T_q\mathcal{M}$, see Figure 5.

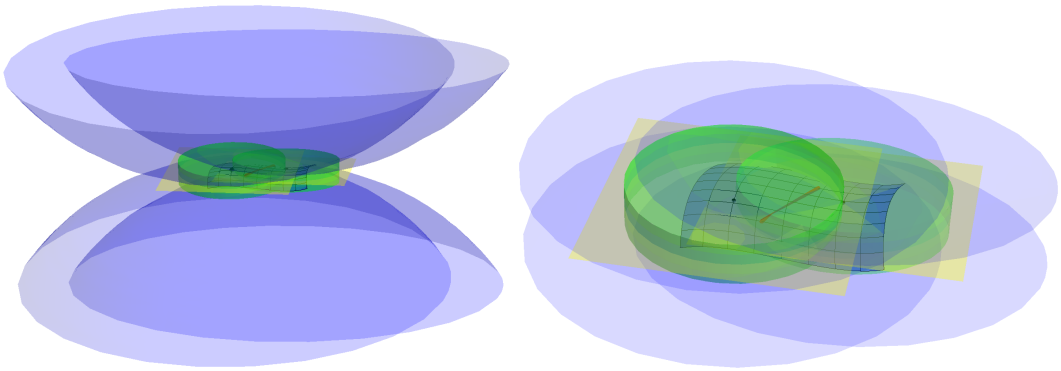


Figure 5 The tangent spaces $T_p\mathcal{M}$ and $T_q\mathcal{M}$ are drawn in yellow. The cylinders $C(T_p\mathcal{M}, r_1, \tilde{r})$ and $C(T_q\mathcal{M}, r_1, \tilde{r})$ are indicated in green. The red line segment lies in both cylinders and therefore its angle with both $T_p\mathcal{M}$ and $T_q\mathcal{M}$ is small.

For the existence of the line segment that is contained in both $C(T_p\mathcal{M}, r_1, \tilde{r})$ and $C(T_q\mathcal{M}, r_1, \tilde{r})$ we need the following corollary of Proposition 13:

▶ **Corollary 19.** *For each $v \in T_pM$ such that $|v-p| < \frac{\sqrt{3}}{2}\mathrm{rch}(\mathcal{M})$ there exists at least one original $\pi_{T_pM}^{-1}(v)$.*

The proof of this statement can be found in the appendix of [10].

▶ **Theorem 20.** *Let $|p-q| \leq \mathrm{rch}(\mathcal{M})/3$, then the angle φ between $T_p\mathcal{M}$ and $T_q\mathcal{M}$ is bounded by*

$$\sin\frac{\varphi}{2} \leq \frac{\left(1-\sqrt{1-\alpha^2}\right)}{\sqrt{\frac{\alpha^2}{4} - (\frac{\alpha^2}{2}+1-\sqrt{1-\alpha^2})^2}} \simeq \alpha + 9\alpha^3/4,$$

where $\alpha = |p-q|/\mathrm{rch}(\mathcal{M})$.

The proof of this result follows the lines as sketched in the overview, and can be found in full in the appendix [10].

▶ Remark 21. The bound we presented above can be tightened by further geometric analysis, in particular by splitting $T_p\mathcal{M}$ into the span of $\pi_{T_p\mathcal{M}}(q) - p$ and its orthocomplement. However we chose to preserve the elementary character of the argument.

With the bound on the angles between the tangent spaces it is not difficult to prove that the projection map is locally a diffeomorphism, as has been done in [19].

References

1. N. Amenta and M. W. Bern. Surface reconstruction by Voronoi filtering. In *SoCG*, pages 39–48, 1998. doi:10.1145/276884.276889.
2. D. Attali, H. Edelsbrunner, and Yu. Mileyko. Weak Witnesses for Delaunay triangulations of Submanifold. In *ACM Symposium on Solid and Physical Modeling*, pages 143–150, Beijing, China, 2007. URL: https://hal.archives-ouvertes.fr/hal-00201055.
3. D. Attali and A. Lieutier. Geometry-driven collapses for converting a Čech complex into a triangulation of a nicely triangulable shape. *Discrete & Computational Geometry*, 54(4):798–825, 2015.
4. M. Belkin, J. Sun, and Y. Wang. Constructing laplace operator from point clouds in $\mathbb{R}^d$. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1031–1040, 2009. doi:10.1137/1.9781611973068.112.
5. J.-D. Boissonnat and F. Cazals. Natural neighbor coordinates of points on a surface. *Computational Geometry Theory & Applications*, 19(2–3):155–173, Jul 2001. URL: http://www.sciencedirect.com/science/article/pii/S0925772101000189.
6. J.-D. Boissonnat, R. Dyer, and A. Ghosh. Constructing intrinsic Delaunay triangulations of submanifolds. Research Report RR-8273, INRIA, 2013. arXiv:1303.6493. URL: http://hal.inria.fr/hal-00804878.
7. J.-D. Boissonnat, R. Dyer, A. Ghosh, and M.H.M.J. Wintraecken. Local criteria for triangulation of manifolds. *Accepted for SoCG 2018*, 2018. URL: hal.inria.fr/hal-01661230.
8. J.-D. Boissonnat and A. Ghosh. Triangulating smooth submanifolds with light scaffolding. *Mathematics in Computer Science*, 4(4):431–461, 2010.
9. J.-D. Boissonnat and S. Oudot. Provably good surface sampling and approximation. In *Symp. Geometry Processing*, pages 9–18, 2003.

10 Jean-Daniel Boissonnat, André Lieutier, and Mathijs Wintraecken. The reach, metric distortion, geodesic convexity and the variation of tangent spaces. full version, 2017. URL: `https://hal.inria.fr/hal-01661227`.

11 S.-W. Cheng, T. K. Dey, and E. A. Ramos. Manifold reconstruction from point samples. In *SODA*, pages 1018–1027, 2005.

12 T. K. Dey. *Curve and Surface Reconstruction: Algorithms with Mathematical Analysis (Cambridge Monographs on Applied and Computational Mathematics)*. Cambridge University Press, New York, NY, USA, 2006.

13 T.K. Dey, J. Giesen, E.A. Ramos, and B. Sadri. Critical points of distance to an ϵ-sampling of a surface and flow-complex-based surface reconstruction. *International Journal of Computational Geometry & Applications*, 18(01n02):29–61, 2008. `doi:10.1142/S0218195908002532`.

14 M. P. do Carmo. *Riemannian Geometry*. Birkhäuser, 1992.

15 H. Federer. Curvature measures. *Trans. Amer. Math. Soc.*, 93(3):418–491, 1959.

16 M. Gromov, M. Katz, P. Pansu, and S. Semmes. *Metric structures for Riemannian and non-Riemannian spaces*. Birkhauser, 2007.

17 K. Menger. Untersuchungen uber allgemeine metrik, vierte untersuchungen zur metrik kurven. *Mathematische Annalen*, 103:466–501, 1930.

18 J. Milnor. *Morse Theory*. Cambridge, 2006.

19 P. Niyogi, S. Smale, and S. Weinberger. Finding the homology of submanifolds with high confidence from random samples. *Discrete & Comp. Geom.*, 39(1-3):419–441, 2008.

20 S. Scholtes. On hypersurfaces of positive reach, alternating Steiner formulae and Hadwiger's Problem. *ArXiv e-prints*, 2013. `arXiv:1304.4179`.

Orthogonal Terrain Guarding is NP-complete

Édouard Bonnet
ENS Lyon, LIP
Lyon, France
edouard.bonnet@dauphine.fr

Panos Giannopoulos
Department of Computer Science, Middlesex University
London, UK
p.giannopoulos@mdx.ac.uk

Abstract

A terrain is an x-monotone polygonal curve, i.e., successive vertices have increasing x-coordinates. Terrain Guarding can be seen as a special case of the famous art gallery problem where one has to place at most k guards on a terrain made of n vertices in order to fully see it. In 2010, King and Krohn showed that Terrain Guarding is NP-complete [SODA '10, SIAM J. Comput. '11] thereby solving a long-standing open question. They observe that their proof does not settle the complexity of Orthogonal Terrain Guarding where the terrain only consists of horizontal or vertical segments; those terrains are called rectilinear or orthogonal. Recently, Ashok et al. [SoCG'17] presented an FPT algorithm running in time $k^{O(k)}n^{O(1)}$ for Dominating Set in the visibility graphs of rectilinear terrains without 180-degree vertices. They ask if Orthogonal Terrain Guarding is in P or NP-hard. In the same paper, they give a subexponential-time algorithm running in $n^{O(\sqrt{n})}$ (actually even $n^{O(\sqrt{k})}$) for the general Terrain Guarding and notice that the hardness proof of King and Krohn only disproves a running time $2^{o(n^{1/4})}$ under the ETH. Hence, there is a significant gap between their $2^{O(n^{1/2}\log n)}$-algorithm and the no $2^{o(n^{1/4})}$ ETH-hardness implied by King and Krohn's result.

In this paper, we answer those two remaining questions. We adapt the gadgets of King and Krohn to rectilinear terrains in order to prove that even Orthogonal Terrain Guarding is NP-complete. Then, we show how their reduction from Planar 3-SAT (as well as our adaptation for rectilinear terrains) can actually be made linear (instead of quadratic).

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases terrain guarding, rectilinear terrain, computational complexity

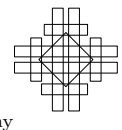
Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.11

Related Version A full version of this paper is available at `https://arxiv.org/abs/1710.00386`

Funding Supported by EPSRC grant FptGeom (EP/N029143/1)

1 Introduction

Terrain Guarding is a natural restriction of the well-known art gallery problem. It asks, given an integer k, and an *x-monotone polygonal chain* or *terrain*, to guard it by placing at most k guards at vertices of the terrain. An *x-monotone polygonal chain* is defined from a sequence of n points of the real plane $\mathbb{R}^2$ $p_1 = (x_1, y_1), p_2 = (x_2, y_2), \ldots, p_n = (x_n, y_n)$ such that $x_1 \leqslant x_2 \leqslant \ldots \leqslant x_n$ as the succession of straight-line edges $p_1p_2, p_2p_3, \ldots, p_{n-1}p_n$. Each point p_i is called a *vertex* of the terrain. We can make each coordinate of the vertices rational without changing the (non-)existence of a solution. We will therefore assume that




the input is given as a list of n pairs of rational numbers, together with the integer k. A point p lying on the terrain is *guarded* (or seen) by a subset S of guards if there is at least one guard $g \in S$ such that the straight-line segment pg is entirely above the polygonal chain. The terrain is said *guarded* if every point lying on the terrain is guarded. The visibility graph of a terrain has as vertices the geometric vertices of the polygonal chain and as edges every pair which sees each other. Again two vertices (or points) see each other if the straight-line segment that they define is above the terrain.

The Orthogonal Terrain Guarding is the same problem restricted to *rectilinear* (also called *orthogonal*) terrains, that is every edge of the terrain is either horizontal or vertical. In other words, p_i and p_{i+1} share the same x-coordinate or the same y-coordinate. We say that a rectilinear terrain is *strictly rectilinear* (or *strictly orthogonal*) if the horizontal and vertical edges alternate, that is, there are no two consecutive horizontal (resp. vertical) edges. Both problems come with two other variants: the *continuous variant*, where the guards can be placed anywhere on the edges of the terrain (and not only at the vertices), and the *graphic variant*, which consists of Dominating Set in the visibility graphs of (strictly rectilinear) terrains. The original problem is sometimes called the *discrete variant*.

It is a folklore observation that for rectilinear terrains, the discrete and continuous variants coincide. Indeed, it is an easy exercise to show that from any feasible solution using guards in the interior of edges, one can move those guards to vertices and obtain a feasible solution of equal cardinality. The only rule to respect is that if an edge, whose interior contained a guard, links a reflex and a convex vertex, then the guard should be moved to the reflex vertex. We will therefore only consider Orthogonal Terrain Guarding and Dominating Set in the visibility graphs of strictly rectilinear terrains. By subdividing the edges of a strictly rectilinear terrain with an at most quadratic number of 180-degree vertices (i.e., vertices incident to two horizontal edges or to two vertical edges), one can make *guarding all the vertices* equivalent to *guarding the whole terrain*. Therefore, Orthogonal Terrain Guarding is not very different from Dominating Set in the visibility graph of (non necessarily strictly) rectilinear terrains (and Terrain Guarding is not very different from Dominating Set in the visibility graph of terrains).

Exponential Time Hypothesis. The *Exponential Time Hypothesis* (usually referred to as the ETH) is a stronger assumption than P≠NP formulated by Impagliazzo and Paturi [14]. A practical (and slightly weaker) statement of ETH is that 3-SAT with n variables cannot be solved in subexponential-time $2^{o(n)}$. Although this is not the original statement of the hypothesis, this version is most commonly used; see also Impagliazzo et al. [15]. The so-called *sparsification lemma* even brings the number of clauses in the exponent.

▶ **Theorem 1** (Impagliazzo and Paturi [14]). *Under the ETH, there is no algorithm solving every instance of* 3-SAT *with n variables and m clauses in time $2^{o(n+m)}$.*

As a direct consequence, unless the ETH fails, even instances with a linear number of clauses $m = \Theta(n)$ cannot be solved in $2^{o(n)}$. Unlike P≠NP, the ETH allows to rule out specific running times. We refer the reader to the survey by Lokshtanov et al. for more information about ETH and conditional lower bounds [23].

Planar satisfiability. Planar 3-SAT was introduced by Lichtenstein [22] who showed its NP-hardness. It is a special case of 3-SAT where the variable/clause incidence graph is planar even if one adds edges between two consecutive variables for a specified ordering of the variables: $x_1, x_2, \ldots, x_n$; i.e., $x_i x_{i+1}$ is an edge (with index $i+1$ taken modulo n). This extra

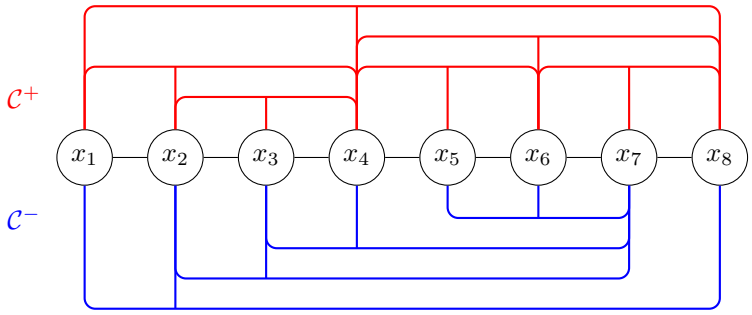


Figure 1 The bipartition $(\mathcal{C}^+, \mathcal{C}^-)$ of a Planar 3-SAT-instance. The three-legged arches represent the clauses. Here is a removal ordering for $\mathcal{C}^-$: remove the clause on x_5, x_6, x_7 and its middle variable x_6, remove the variable x_5, remove the clause on x_3, x_4, x_7 and its middle variable x_4, remove the clause on x_2, x_3, x_7 and its middle variable x_3, remove the variable x_7, remove the clause x_1, x_2, x_8 and its middle variable x_2.

structure makes this problem particularly suitable to reduce to planar or geometric problems. As a counterpart, the ETH lower bound that one gets from a linear reduction from Planar 3-SAT is worse than with a linear reduction from 3-SAT; it only rules out a running time $2^{o(\sqrt{n})}$. Indeed, Planar 3-SAT can be solved in time $2^{O(\sqrt{n})}$ and, unless the ETH fails, cannot be solved in time $2^{o(\sqrt{n})}$. A useful property of any Planar 3-SAT-instance is that its set of clauses $\mathcal{C}$ can be partitioned into $\mathcal{C}^+$ and $\mathcal{C}^-$ such that both $\mathcal{C}^+$ and $\mathcal{C}^-$ admit a *removal ordering*. A *removal ordering* is a sequence of the two following deletions:

- (a) removing a variable which is not present in any remaining clause
- (b) removing a clause on *three consecutive remaining variables* together with the *middle variable*

which ends up with an empty set of clauses. By *three consecutive remaining variables*, we mean three variables x_i, x_j, x_k, with $i < j < k$ such that $x_{i+1}, x_{i+2}, \ldots, x_{j-1}$ and $x_{j+1}, x_{j+2}, \ldots, x_{k-1}$ have all been removed already. The middle variable of the clause is x_j. For an example, see Figure 1.

Order claim. The following visibility property in a terrain made King and Krohn realize that they will crucially need the extra structure given by the planarity of 3-SAT-instances.

▶ **Lemma 2** (Order Claim, see Figure 2). *If a, b, c, d happen in this order from the left endpoint of the terrain to its right endpoint, a and c see each other, and b and d see each other, then a and d also see each other.*

In particular, this suggests that checking in the terrain *if a clause is satisfied* can only work if the encodings of the three variables contained in the clause are *consecutive*.

Related work and remaining open questions for terrain guarding. Terrain Guarding was shown NP-hard [18] and can be solved in time $n^{O(\sqrt{k})}$ [1]. This contrasts with the

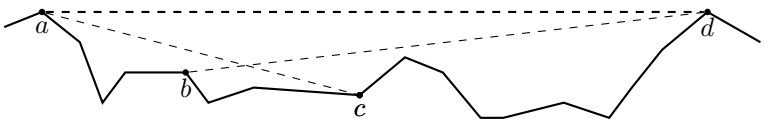


Figure 2 The order claim.

parameterized complexity of the more general art gallery problem where an algorithm running in time $f(k)n^{o(k/\log k)}$ for any computable function f would disprove the ETH, both for the variant Point Guard Art Gallery where the k guards can be placed anywhere inside the gallery (polygon with n vertices) and for the variant Vertex Guard Art Gallery where the k guards can only be placed at the vertices of the polygon [4], even when the gallery is a simple polygon (i.e., does not have holes). Dominating Set on the visibility graph of strictly rectilinear terrains can be solved in time $k^{O(k)}n^{O(1)}$ [1], while it is still not known if (Orthogonal) Terrain Guarding can be solved in FPT time $f(k)n^{O(1)}$ with respect to the number of guards.

There has been a succession of approximation algorithms with better and better constant ratios [16, 7, 2, 13]. Eventually, a PTAS was found for Terrain Guarding (hence for Orthogonal Terrain Guarding) [20] using local search and an idea developed by Chan and Har-Peled [6] and Mustafa and Ray [24] which consists of applying the planar separator theorem to a (planar) graph relating local and global optima. Interestingly, this planar graph is the starting point of the subexponential algorithm of Ashok et al. [1].

Again the situation is not nearly as good for the art gallery problem. If holes are allowed in the polygon, the main variants of the art gallery problem are as hard as the Set Cover problem; hence a $o(\log n)$-approximation cannot exist unless P=NP [11]. Eidenbenz also showed that a PTAS is unlikely in simple polygons [10]. For simple polygons, there is a $O(\log \log OPT)$-approximation [17, 19] for Vertex Guard Art Gallery, using the framework of Brönnimann and Goodrich to transform an ε-net finder into an approximation algorithm, and for Point Guard Art Gallery there is a randomized $O(\log OPT)$-approximation under some mild assumptions [5], building up on [9, 8]. If a small fraction of the polygon can be left unguarded there is again a $O(\log OPT)$-approximation [12]. A constant approximation is known for *monotone polygons* [21], where a monotone polygon is made of two terrains sharing the same left and right endpoints and except those two points the two terrains are never touching nor crossing.

The classical complexity of Orthogonal Terrain Guarding remains the most pressing open question [1].

▶ **Open question 1.** *Is* Orthogonal Terrain Guarding *in P or NP-hard?*

In the conclusion of the paper by Ashok et al. [1], the authors observe that the construction of King and Krohn [18] rules out for Terrain Guarding a running time of $2^{o(n^{1/4})}$, under the ETH. Indeed the reduction from Planar 3-SAT (which is not solvable in time $2^{o(\sqrt{n})}$ unless the ETH fails) and its adaptation for Orthogonal Terrain Guarding in the current paper have a quadratic blow-up: the terrain is made of $\Theta(m) = \Theta(n)$ chunks containing each $O(n)$ vertices. On the positive side, the subexponential algorithm of Ashok et al. runs in time $2^{O(\sqrt{n}\log n)}$ [1]. Therefore, there was still a significant gap between the algorithmic upper and lower bounds.

▶ **Open question 2.** *Assuming the ETH, what is the provably best asymptotic running time for* Terrain Guarding *and* Orthogonal Terrain Guarding*?*

We resolve both open questions. It is remarkable that within the last ten years, knowledge on the computational complexity of (Orthogonal) Terrain Guarding has gone from a handful of constant-approximation algorithms and no lower bound at all to tight approximability under P≠NP and almost tight ETH-hardness. Our paper provides the two missing pieces: the NP-hardness of Orthogonal Terrain Guarding and a versatile refinement of a quadratic reduction from Planar 3-SAT to a linear reduction.

Organization. In Section 2, we address Open question 1 by showing that Orthogonal Terrain Guarding is also NP-hard. We design a rectilinear subterrain with a constant number of vertices which simulates a triangular pocket surrounded by two horizontal segments. We can then adapt the reduction of King and Krohn [18] to rectilinear terrains. Our orthogonal gadgets make an extensive use of the triangular pockets.

In Section 3, we show how to make both reductions from Planar 3-SAT linear instead of quadratic. This shows that, under the ETH, Terrain Guarding and Orthogonal Terrain Guarding cannot be solved in $2^{o(\sqrt{n})}$, and thereby resolve Open question 2. Unless the ETH fails, the $2^{O(\sqrt{n}\log n)}$-time algorithm of Ashok et al. is optimal up to logarithmic factors in the exponent.

2 Orthogonal Terrain Guarding is NP-complete

King and Krohn give a reduction with a quadratic blow-up from Planar 3-SAT to Terrain Guarding [18]. They argue that the order claim entails some critical obstacle against straightforward hardness attempts. In some sense, the subexponential algorithm running in time $n^{O(\sqrt{n})}$ of Ashok et al. [1] proves them right: unless the ETH fails, there cannot be a linear reduction from 3-SAT to Terrain Guarding. It also justifies their idea of starting from the planar variant of 3-SAT. Indeed, this problem can be solved in time $2^{O(\sqrt{n})}$. However, we will see that the quadratic blow-up of their construction is avoidable. In the next section, we show how to make their reduction (and ours for the orthogonal case) linear. In this section, we focus on our main result: the NP-hardness of Orthogonal Terrain Guarding.

From far, King and Krohn's construction looks like a V-shaped terrain. If one zooms in, one perceives that the V is made of $\Theta(n)$ connected subterrains called *chunks*. If one zooms a bit more, one sees that the chunks are made of up to n variable encodings each. Let us order the chunks from bottom to top; in this order, the chunks alternate between the right and the left of the V (see Figure 3).

The construction is such that only two consecutive chunks interact. More precisely, a vertex of a given chunk T_i only sees bits of the terrain contained in T_{i-1}, T_i, and T_{i+1}. Half-way to the top is the chunk T_0 that can be seen as the *initial* one. It contains the

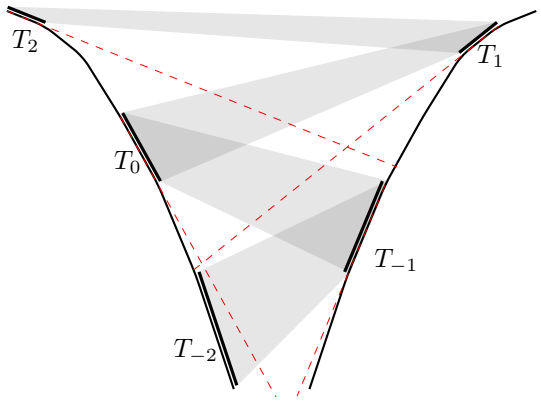


Figure 3 The V-shaped terrain and its ordered chunks. The chunk T_i only sees parts of chunks T_{i-1} and T_{i+1}. The *initial* chunk T_0 contains an encoding of each variable. Below this level (chunks with a negative index), we will check the clauses of $\mathcal{C}^-$. Above this level (chunks with a positive index), we will deal with the clauses of $\mathcal{C}^+$.

encoding of *all* the variables of the Planar 3-SAT-instance. Concretely, the reasonable choices to place guards on the chunk T_0 are interpreted as setting each variable to either *true* or *false*. Let $(\mathcal{C}^+, \mathcal{C}^-)$ be the bipartition of the clauses into two sets with a removal ordering for the variables ordered as $x_1, x_2, \ldots, x_n$. Let $C_1^+, C_2^+, \ldots, C_s^+$ (resp. $C_1^-, C_2^-, \ldots, C_{m-s}^-$) be the order in which the clauses of $\mathcal{C}^+$ (resp. $\mathcal{C}^-$) disappear in this removal ordering. Every chunk below T_0, i.e., with a negative index, are dedicated to checking the clauses of $\mathcal{C}^-$ in the order $C_1^-, C_2^-, \ldots, C_{m-s}^-$, while every chunk above T_0, i.e., with a positive index, will check *if the clauses of* C^+ *are satisfied* in the order $C_1^+, C_2^+, \ldots, C_s^+$. The placement of the chunks will *propagate downward and upward* the truth assignment of T_0, and simulate the operations of a removal ordering: checking/removing a clause and its middle variable, removing a useless variable. Note that for those gadgets, we will have to distinguish if we are *going up* ($\mathcal{C}^+$) or *going down* ($\mathcal{C}^-$). In addition, the respective position of the positive and negative literals of a variable appearing in the next clause to check will matter. So, we will require a gadget to invert those two literals if needed.

To sum up, the reduction comprises the following gadgets: encoding a variable (variable gadget), propagation of its assignment from one chunk to a consecutive chunk (interaction of two variable gadgets), inverting its two literals (inverter), checking a clause *upward* and removing the henceforth useless middle variable (upward clause gadget), checking a clause *downward* and removing the henceforth useless middle variable (downward clause gadget), removing a variable (upward/downward deletion gadget). Although King and Krohn rather crucially rely on having different slopes in the terrain, we will mimic their construction gadget by gadget with an orthogonal terrain. We start by showing how to simulate a restricted form of a triangular pocket. This will prove to be a useful building block.

The simulation of a *right trapezoid pocket* giving rise to the *right triangular pocket* is depicted on Figure 4. The idea is that the vertex p at the bottom of the pit is only seen by four vertices (no vertex outside this gadget will be able to see p). Among those four vertices, u sees a strict superset of what the others see. Hence, we can assume with no loss of generality that a guard is placed on u. Now, u sees the part of the terrain represented in bold. Even if vertex u sees a part of the vertical edge incident to a (actually the construction could avoid it), this information can be discarded since the guard responsible for seeing a will see this edge entirely. Everything is therefore equivalent to guarding the terrain with the right trapezoid pocket drawn in the middle of Figure 4 with a budget of guards decreased

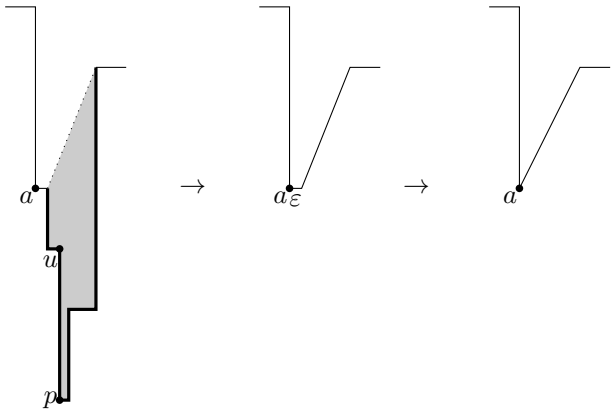


Figure 4 Simulation of a right trapezoid pocket and a right triangular pocket. The right triangular pocket is obtained from the right trapezoid by making the distance ε sufficiently small.

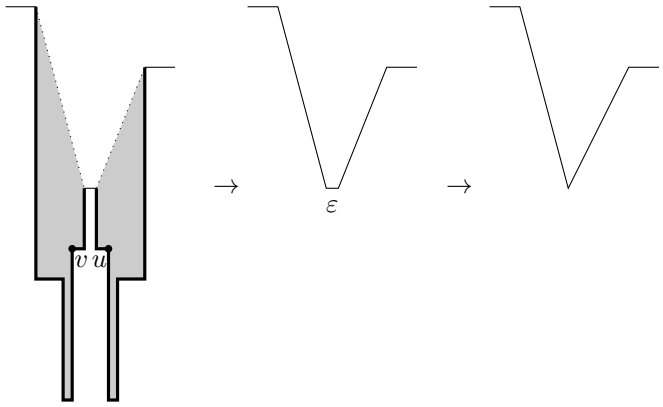


Figure 5 Simulation of a trapezoid pocket and a triangular pocket. The triangular pocket is obtained from the trapezoid by making the distance ε arbitrary small.

by one. If the length of the horizontal edge incident to a is made small enough, then every guard seeing a will see the whole edge, thereby simulating the right triangular pocket to the right of the figure.

The acute angle made by the right triangular pocket and the altitude of the leftmost and rightmost horizontal edge in this gadget can be set at our convenience. We will represent triangular pockets in the upcoming gadgets. The reader should keep in mind that they are actually a shorthand for a rectilinear subterrain.

With the same idea, one can simulate a general triangular pocket as depicted on Figure 5, with the budget decreased by two guards. Again, the non-reflex angle made by the triangular pocket and the altitude of the leftmost and rightmost horizontal edges can be freely chosen. The reason why those triangular pockets do not provide a straightforward reduction from the general Terrain Guarding problem is that the pocket has to be preceded and succeeded by horizontal edges.

The variable gadget is depicted on Figure 6. It is made of three right triangular pockets. Placing a guard on v_x (resp. $v_{\overline{x}}$) is interpreted as setting the variable x to *true* (resp. *false*).

The propagation of a variable assignment from one chunk to the next chunk is represented on Figure 7. On all the upcoming figures, we adopt the convention that red dashed lines materialize a blocked visibility (the vertex cannot see anything *below this line*) and black dashed lines highlight important visibility which sets apart the vertex from other vertices.

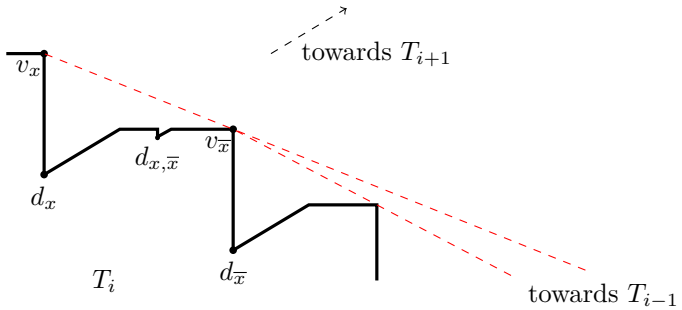


Figure 6 A variable gadget. We omit superscript i on all the labels. Placing a guard at vertex v_x to see d_x corresponds to setting variable x to true, while placing it at vertex $v_{\overline{x}}$ to see $d_{\overline{x}}$ corresponds to setting x to false. Both v_x^{i+1} and $v_{\overline{x}}^{i+1}$ of T_{i+1} (not represented on this picture) see $d_{x,\overline{x}}$ of T_i.

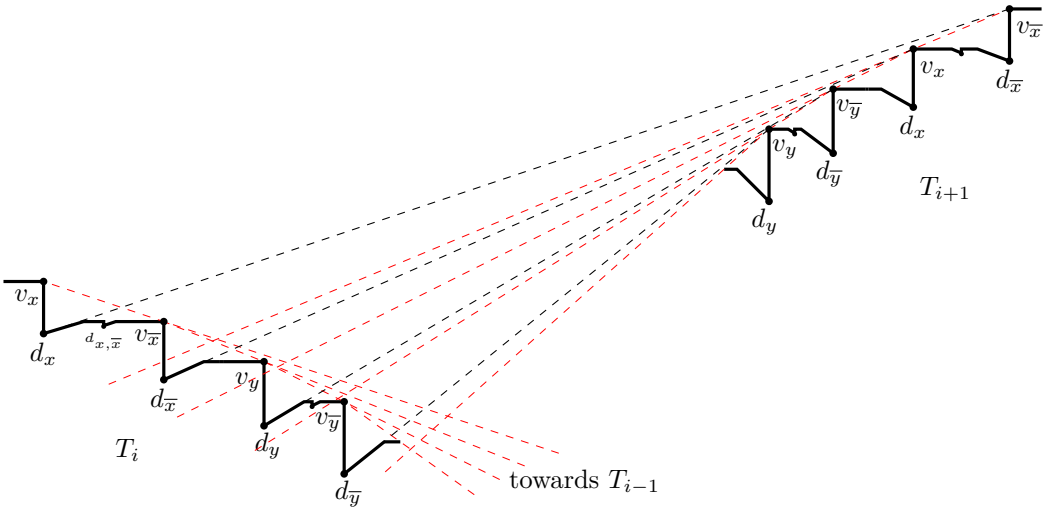


Figure 7 Propagating variable assignments upward and downward. Note that the positive literal alternates being above or below the negative literal. We represent two variables x and y to illustrate how the corresponding gadgets are not interfering.

Say, one places a guard at vertex $v^i_{\overline{x}}$ to see (among other things) the vertex $d^i_{\overline{x}}$. Now, d^i_x and $d^i_{x,\overline{x}}$ remain to be seen. The only way of guarding them with one guard is to place it at vertex $v^{i+1}_{\overline{x}}$. Indeed, only vertices on the chunk T_{i+1} can possibly see both. But the vertices higher than $v^{i+1}_{\overline{x}}$ cannot see them because their visibility is blocked by $v^{i+1}_{\overline{x}}$ or a vertex to its right, while the vertices lower than $v^{i+1}_{\overline{x}}$ are too low to see the very bottom of those two triangular pockets. The same mechanism (too high $\rightarrow$ blocked visibility, too low $\rightarrow$ too flat angle) is used to ensure that the different variables do not interfere.

Symmetrically, the only vertex seeing both $d^i_{x,\overline{x}}$ and $d^i_{\overline{x}}$ is v^{i+1}_x. So, placing a guard at v^i_x forces to place the other guard at v^{i+1}_x. The chosen literal goes from being above (resp. below) in chunk T_i to being below (resp. above) in chunk T_{i+1}. Each d-vertex (i.e., vertex of the form $d^\bullet_\bullet$) has its visibility dominated in the one of a v-vertex (of the form $v^\bullet_\bullet$). Indeed, the visibility of d^i_x and $d^i_{\overline{x}}$ is contained in the visibilty of v^i_x and $v^i_{\overline{x}}$, respectively, while $d^i_{x,\overline{x}}$ has its visibility dominated by the one of v^{i+1}_x or $v^{i+1}_{\overline{x}}$ (what a vertex sees from below in a rectilinear terrain is irrelevant). Each non v-vertex has its visibility contained in the one of a v-vertex. Seeing the d-vertices with v-vertices is enough to see the entire subterrain/chunk. Thus, the problem can be seen as a red-blue domination: taking v-vertices (red) to dominate the d-vertices (blue). The red-blue visibility graph corresponding to the propagation of variable assignments is shown on Figure 8. The only way of guarding the $3z$ d-vertices on chunk T^i (corresponding to z variables) with a budget of z guards on T^i and z guards on T^{i+1} is to place z guards on v-vertices of chunk T_i and z guards on v-vertices of chunk T_{i+1} in a consistent way: the assignment of each variable is preserved.

We also need an alternative way of propagating truth assignments such that the chosen literal stays above or stays below on its respective chunk. This gadget is called *inverter*. It requires an extra guard compared to the usual propagation. The inverter gadget allows us to position the three literals of the clause to check and delete at the right spots.

It consists of a right triangular pocket whose bottom vertex is $d^i_{x,\overline{x}}$ surrounded by two rectangular pockets whose bottom vertices e^i_x, f^i_x and $e^i_{\overline{x}}, f^i_{\overline{x}}$ are only seen among the v-vertices by v^{i+1}_x, v^i_x and $v^{i+1}_{\overline{x}}, v^i_{\overline{x}}$, respectively. On top of the rectangular pockets, g^i_x sees both $e^i_{\overline{x}}$ and

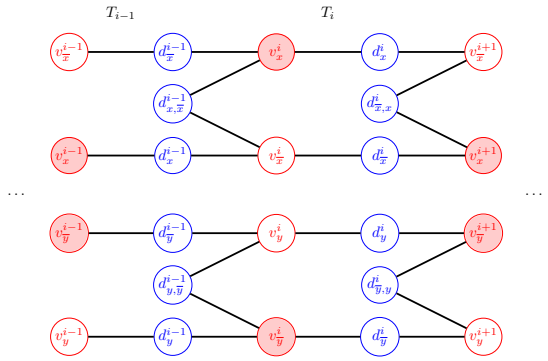


Figure 8 The red-blue domination graph for variable-assignment propagation.

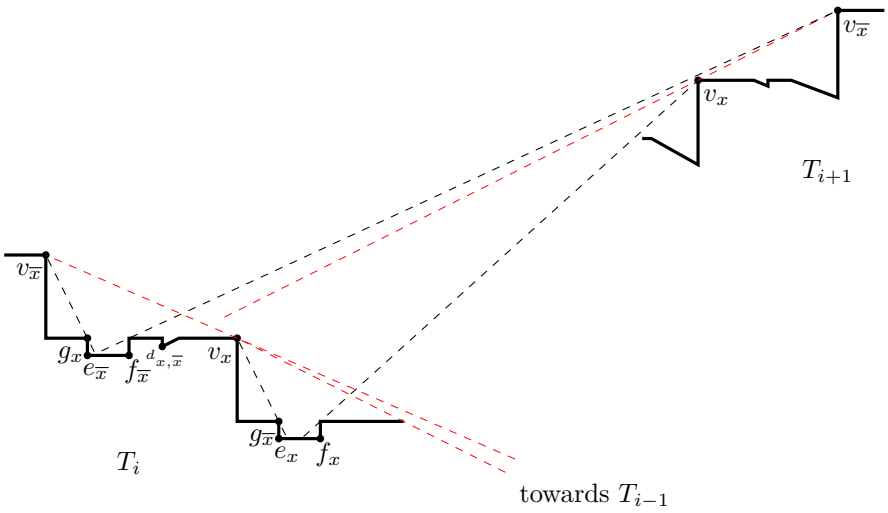


Figure 9 The inverter gadget. We omit the superscripts i and $i+1$. If a guard should be placed on at least one vertex among $v_{\overline{x}}^{\ell}$ and v_x^{ℓ} (for $\ell \in \{i, i+1\}$), then the two ways of seeing the four vertices $e_{\overline{x}}^i$, $f_{\overline{x}}^i$, e_x^i, f_x^i with three guards are $\{v_{\overline{x}}^i, g_{\overline{x}}^i, v_{\overline{x}}^{i+1}\}$ and $\{v_x^i, g_x^i, v_x^{i+1}\}$.

$f_{\overline{x}}^i$, whereas $g_{\overline{x}}^i$ sees both e_x^i and f_x^i. Actually, g_ℓ^i is only one of the four vertices seeing both e_ℓ^i and f_ℓ^i (which includes e_ℓ^i and f_ℓ^i themselves). We choose g_ℓ^i as a representative of this class. What matters to us is that the four vertices seeing both e_ℓ^i and f_ℓ^i do not see anything more than the rectangular pocket; the other parts of the terrain that they might guard are seen by any v-vertex on chunk T_{i+1} anyway.

The pockets are designed so that v_x^i and v_x^{i+1} (resp. $v_{\overline{x}}^i$ and $v_{\overline{x}}^{i+1}$) together see the whole edge $e_x^i f_x^i$ (resp. $e_{\overline{x}}^i f_{\overline{x}}^i$) and therefore the entire pocket. Again, the only two v-vertices to see $d_{x,\overline{x}}^i$ are v_x^{i+1} and $v_{\overline{x}}^{i+1}$. The e- and f-vertices are added to the blue vertices and the g-vertices are added to the red vertices, since the latter sees more than the former, and since seeing the e- and f-vertices are sufficient to also see the g-vertices. The red-blue domination graph is depicted on Figure 10.

Guarding $d_{x,\overline{x}}^{i-1}$ (resp. guarding $d_{x,\overline{x}}^i$) requires to take one v-vertex among $v_x^i, v_{\overline{x}}^i$ (resp. v_x^{i+1}, $v_{\overline{x}}^{i+1}$). Note that if one makes two inconsistent choices such as placing guards at v_x^i and $v_{\overline{x}}^{i+1}$ (or $v_{\overline{x}}^i$ and v_x^{i+1}), then it is not possible to see both rectangular pockets with one extra guard. Whereas, placing three guards at v_x^i, g_x^i, v_x^{i+1} or $v_{\overline{x}}^i, g_{\overline{x}}^i, v_{\overline{x}}^{i+1}$ would cover both rectangular pockets; hence the propagation of the truth assignment.

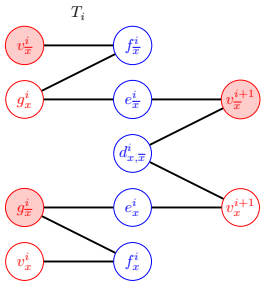


Figure 10 The red-blue domination graph for the inverter gadget.

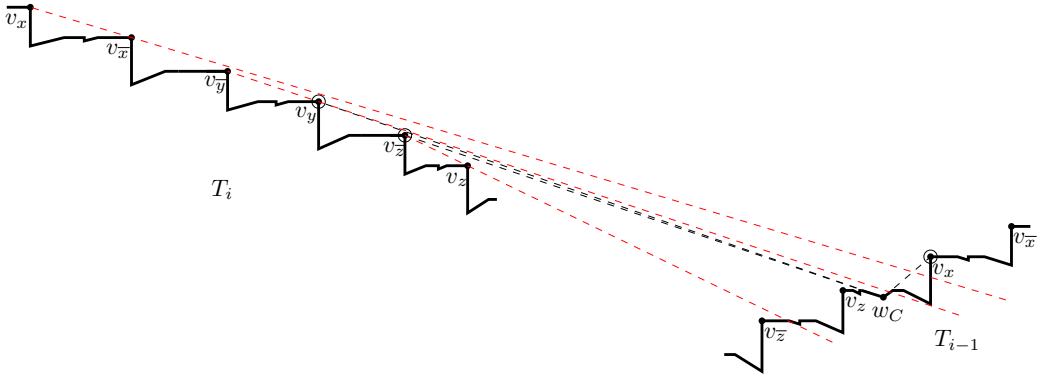


Figure 11 The downward clause gadget for $C = x \vee y \vee \neg z$. We use the usual propagation for variables x and z. Variable y disappears from T_{i-1} and downward. The inverters have been used to place, on T_i, the literals of C at positions 1, 4, and 5. Vertex w_C is seen only by v_y^i, $v_{\overline{z}}^i$, and v_x^{i-1} (circled); hence it is seen if and only if the chosen assignment satisfies C.

So far, the gadgets that we presented can be used *going up* along the chunks of positive index as well as *going down* along the chunks of negative index. For the clause gadgets, we will have to distinguish the *downward clause gadget* when we are below T_0 (and going down) and the *upward clause gadget* when we are above T_0 (and going up). The reason we cannot design a single gadget for both situations is that the middle variable which needs be deleted is in one case, in the lower chunk, and in the other case, in the higher chunk.

To check a clause downward on three consecutive variables x, y, z, we place on chunk T_i, thanks to a preliminary use of inverter gadgets, the three literals satisfying the clause at the relative positions 1, 4, and 5 when the six literals of x, y, z are read from top to bottom. Figure 11 shows the downward clause gadget for the clause $x \vee y \vee \neg z$. On chunk T_{i-1} just below, we find the usual encoding of variables x and z, which propagates the truth assignment of those two variables. The variable gadget of y is replaced by the right triangular pocket whose bottom is $d_{y,\overline{y}}^{i-1}$, and a general triangular pocket whose bottom w_C is only seen by the v-vertices $v_{\ell_1}^{i-1}$ (on chunk T_{i-1}), and $v_{\ell_2}^i$ and $v_{\ell_3}^i$ (on chunk T_i), with $C = \ell_1 \vee \ell_2 \vee \ell_3$. On chunk T_{i-1} and below, no v-vertex corresponding to variable y can be found.

Hence, vertex w_C is only guarded if the choices of the guards at the v-vertices correspond to an assignment satisfying C. The terrain visible to w_C is also covered by $v_{\ell_1}^{i-1}$, hence it is a blue vertex. The red-blue domination graph associated to a downward clause is represented on Figure 12.

To check a clause upward on three consecutive variables x, y, z, we place on chunk T_i, thanks to a preliminary use of inverter gadgets, the three literals satisfying the clause at the

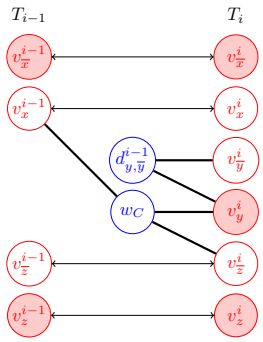


Figure 12 The red-blue domination graph for the downward clause gadget for $C = x \vee y \vee \neg z$. The double arcs symbolize that, due to the propagator, the variable-assignment of x and z should be the same between T_i and T_{i-1}. The only assignment that does not dominate w_C is $\overline{x}$, $\overline{y}$, z, as it should.

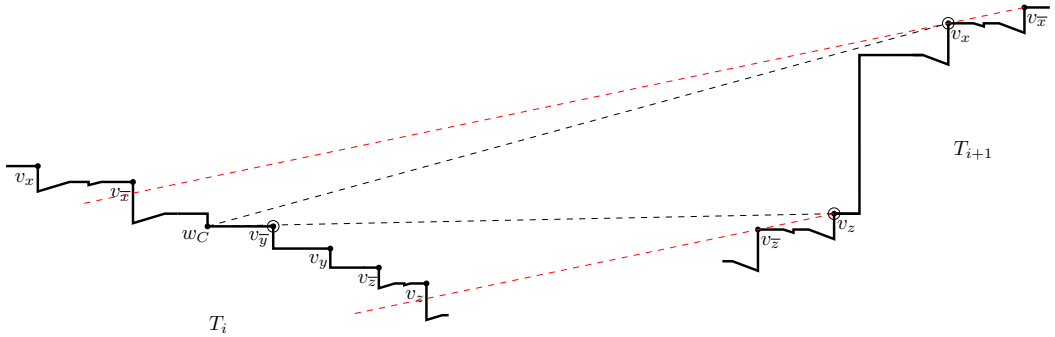


Figure 13 The upward clause gadget for $C = x \vee \neg y \vee z$. We use the usual propagation for variables x and z. Variable y disappears from T_{i+1} and upward. The inverters have been used to place, on T_i, the literals of C at positions 1, 3, and 6. Vertex w_C is seen only by $v_{\overline{y}}^i$, v_x^{i+1}, and v_z^{i+1} (circled); hence it is seen if and only if the chosen assignment satisfies C.

relative positions 1, 3, and 6 when the six literals of x, y, z are read from top to bottom. We exclude the three right triangular pockets for the encoding of the middle variable y. At the same altitude as the v-vertex corresponding to the literal of y satisfying the clause, we have a designated vertex w_C. On the chunk T_{i+1}, we find the usual encoding of variables x and z, which propagates the truth assignment of those two variables, but the encoding of variable y is no longer present (in this chunk and in all the chunks above). Figure 13 shows the upward clause gadget for the clause $x \vee \neg y \vee z$.

Vertex w_C is only seen by the v-vertices $v_{\ell_2}^i$ (on chunk T_i) and $v_{\ell_1}^{i+1}$ and $v_{\ell_3}^{i+1}$ (on chunk T_{i+1}), where $C = \ell_1 \vee \ell_2 \vee \ell_3$. The particularity of two consecutive chunks encoding an upward clause gadget is that T_i is not entirely below T_{i+1}. In fact, all the encodings of variables above y on chunk T_{i+1} are above all the encodings of variables above y on chunk T_i. The latter are above all the encodings of variables below y on chunk T_{i+1}, which are, in turn, above all the encodings of variables below y on chunk T_i. Again, vertex w_C is only guarded if the choices of the guards at the v-vertices correspond to an assignment satisfying C, as depicted in Figure 14.

Finally, we design upward and downward variable deletion gadgets; the description of these gadgets can be found in the full version [3]. The reader can just think of them as simplifications of the clause gadgets where vertex w_c is suppressed. This ends the list of gadgets.

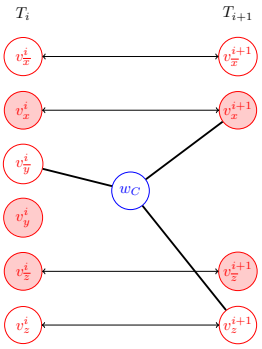


Figure 14 The red-blue domination graph for the upward clause gadget for $C = x \vee \neg y \vee z$. The double arcs symbolize that, due to the propagator, the variable-assignment of x and z should be the same between T_i and T_{i+1}. The only assignment that does not dominate w_C is $\overline{x}$, y, $\overline{z}$, as it should.

The gadgets are assembled as in the reduction of King and Krohn. From the initial chunk T_0 and going up (resp. going down), one realizes step by step (chunk by chunk) the elementary operations to check the clauses of $\mathcal{C}^+$ (resp. $\mathcal{C}^-$) in the order $C_1^+, C_2^+, \dots C_s^+$ (resp. $C_1^-, C_2^-, \dots C_{m-s}^-$) including propagation, inversion of literals, upward clause checking (resp. downward clause checking), and upward variable deletion (resp. downward variable deletion). Each chunk has $O(n)$ vertices. Each clause takes $O(1)$ chunks to be checked. So the total number of chunks is $O(m) = O(n)$ and the total number of vertices is $O(n^2)$.

We call *total budget* the total number of guards allowed. The total budget is fixed as one per right triangular pocket, two per general triangular pocket, one per variable encoding (including the slightly different one at inverters and the one just before an upward deletion), and one extra per inverter. Note that the lone $d^{\bullet}_{x,\overline{x}}$ in a downward clause gadget or a downward deletion gadget does not count as a variable gadget and does not increase the budget. To give an unambiguous definition of the number of variable encodings, we count the number of pairs i, x such that the vertices v_x^i and $v_{\overline{x}}^i$ exist.

We explained why the guards inside the triangular pockets can be placed (and the budget reduced). The correctness of the reduction is similar to King and Krohn's. The d-vertices force the placement of at least one guard in each variable encoding. We argued this to be sufficient to see all the right triangular and rectangular pockets if and only if the variable assignments are consistent between two consecutive chunks (by completing with guards g_ℓ^i at each inverter where ℓ is the literal chosen to be true). The terrain is entirely seen whenever the m general triangular pockets corresponding to the m clauses are all guarded, which happens if and only if the *truth assignment* chosen on chunk T_0 satisfies all the clauses.

This shows that Orthogonal Terrain Guarding and Dominating Set on visibility graph of rectilinear terrains are NP-hard. Recall that the continuous variant of Orthogonal Terrain Guarding is equivalent to its discrete counterpart. Membership in NP of all those variants is therefore trivial. What is left to prove is that Dominating Set on the visibility graph of *strictly* rectilinear terrains is NP-hard. Our reduction almost directly extends to this variant. The only issue is with the general triangular pocket gadget. Indeed, when the two guards are placed inside the pocket, all the internal vertices are guarded. In Orthogonal Terrain Guarding, one still needed to see the interior of the tiny top horizontal edge. This is no longer required in Dominating Set. Observe that the general triangular pocket is only used in the downward clause gadget. In the full version [3], we explain how we can make the downward clause gadget without the general triangular pocket.

3 ETH-Hardness of (Orthogonal) Terrain Guarding

We now explain how to turn those quadratic reductions into linear reductions by taking a step back. This step back is the reduction from 3-SAT to Planar 3-SAT by Lichtenstein [22], or rather, the instances of Planar 3-SAT it produces. The idea of Lichtenstein in this classic paper is to replace each intersection of a pair of edges in the incidence graph of the formula by a constant-size planar gadget, called crossover gadget. Using the sparsification of Impagliazzo et al. [15], even instances of 3-SAT with a linear number of clauses cannot be solved in subexponential time, under the ETH. Hence, the number of edges in the incidence graph of the formula can be assumed to be linear in the number N of variables. Thus there are at most a quadratic number $n = \Theta(N^2)$ of intersections; which implies a replacement of the intersections by a quadratic number n of constant-size crossover gadgets. We cannot expect to improve over the reduction of Lichtenstein since there is a matching algorithm solving Planar 3-SAT in time $2^{O(\sqrt{n})}$.

What we will do instead is to reduce the number of chunks that we actually need *and* to get a better upper bound of the number of vertices in a large fraction of the chunks. In the reduction by King and Krohn, each single clause incurs a constant number of chunks: to place the literals at the right position and to check the clause. The only requirement for a clause to be checked is that it operates on consecutive variables (discarding the deleted variables of the linear order). Therefore, nothing prevents us from checking several clauses *in parallel* if they happen to be on disjoint and consecutive variables.

A first observation is that the $\Theta(n)$ clauses of the crossover gadgets can be checked in parallel with only $O(1)$ chunks. Indeed, the constant number of clauses within each crossover gadget operates on pairwise-disjoint sets of variables. A second observation is that in all the remaining chunks only $\Theta(N)$ variables and $\Theta(N)$ clauses are left to be checked: they correspond to the original variables and clauses of the sparse 3-SAT instances. Therefore, the total number of vertices needed for the terrain is $O(1) \times \Theta(n) + \Theta(N) \times \Theta(N) = \Theta(n) + \Theta(N^2) = \Theta(n)$.

Thus, assuming the ETH, the $2^{O(\sqrt{n}\log n)}$ algorithm for guarding terrains [1] is optimal up to the logarithmic factor in the exponent for both Orthogonal Terrain Guarding and Terrain Guarding.

4 Perspectives

We have shown that Orthogonal Terrain Guarding is NP-complete, as well as its variants. We have presented a generic way of tightening quadratic reductions from Planar 3-SAT to linear. This applies to Terrain Guarding and Orthogonal Terrain Guarding and establishes that the existing $2^{\tilde{O}(\sqrt{n})}$-time algorithm is essentially optimal under the ETH, up to logarithmic factors in the exponent.

The principal remaining open questions concern the parameterized complexity of terrain guarding.

- (1) Is Terrain Guarding FPT parameterized by the number of guards?
- (2) Is Orthogonal Terrain Guarding FPT parameterized by the number of guards?

References

1 Pradeesha Ashok, Fedor V. Fomin, Sudeshna Kolay, Saket Saurabh, and Meirav Zehavi. Exact algorithms for terrain guarding. In *33rd International Symposium on Computational*

Geometry, SoCG 2017, July 4-7, 2017, Brisbane, Australia, pages 11:1–11:15, 2017. doi:10.4230/LIPIcs.SoCG.2017.11.

2 Boaz Ben-Moshe, Matthew J. Katz, and Joseph S. B. Mitchell. A constant-factor approximation algorithm for optimal 1.5d terrain guarding. *SIAM J. Comput.*, 36(6):1631–1647, 2007. doi:10.1137/S0097539704446384.

3 Édouard Bonnet and Panos Giannopoulos. Orthogonal terrain guarding is NP-complete. *CoRR*, abs/1710.00386, 2017. arXiv:1710.00386.

4 Édouard Bonnet and Tillmann Miltzow. Parameterized hardness of art gallery problems. In *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*, pages 19:1–19:17, 2016. doi:10.4230/LIPIcs.ESA.2016.19.

5 Édouard Bonnet and Tillmann Miltzow. An approximation algorithm for the art gallery problem. In *33rd International Symposium on Computational Geometry, SoCG 2017, July 4-7, 2017, Brisbane, Australia*, pages 20:1–20:15, 2017. doi:10.4230/LIPIcs.SoCG.2017.20.

6 Timothy M. Chan and Sariel Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. *Discrete & Computational Geometry*, 48(2):373–392, 2012. doi:10.1007/s00454-012-9417-5.

7 Kenneth L. Clarkson and Kasturi R. Varadarajan. Improved approximation algorithms for geometric set cover. *Discrete & Computational Geometry*, 37(1):43–58, 2007. doi:10.1007/s00454-006-1273-8.

8 Ajay Deshpande, Taejung Kim, Erik D. Demaine, and Sanjay E. Sarma. A pseudopolynomial time O (log n)-approximation algorithm for art gallery problems. In *Algorithms and Data Structures, 10th International Workshop, WADS 2007, Halifax, Canada, August 15-17, 2007, Proceedings*, pages 163–174, 2007. doi:10.1007/978-3-540-73951-7_15.

9 Alon Efrat and Sariel Har-Peled. Guarding galleries and terrains. *Inf. Process. Lett.*, 100(6):238–245, 2006. doi:10.1016/j.ipl.2006.05.014.

10 Stephan Eidenbenz. Inapproximability results for guarding polygons without holes. In *Algorithms and Computation, 9th International Symposium, ISAAC '98, Taejon, Korea, December 14-16, 1998, Proceedings*, pages 427–436, 1998. doi:10.1007/3-540-49381-6_45.

11 Stephan Eidenbenz, Christoph Stamm, and Peter Widmayer. Inapproximability results for guarding polygons and terrains. *Algorithmica*, 31(1):79–113, 2001. doi:10.1007/s00453-001-0040-8.

12 Khaled Elbassioni. Finding small hitting sets in infinite range spaces of bounded vc-dimension. In *33rd International Symposium on Computational Geometry, SoCG 2017, July 4-7, 2017, Brisbane, Australia*, pages 40:1–40:15, 2017. doi:10.4230/LIPIcs.SoCG.2017.40.

13 Khaled M. Elbassioni, Erik Krohn, Domagoj Matijevic, Julián Mestre, and Domagoj Severdija. Improved approximations for guarding 1.5-dimensional terrains. *Algorithmica*, 60(2):451–463, 2011. doi:10.1007/s00453-009-9358-4.

14 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *Journal of Computer and System Sciences*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

15 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

16 James King. A 4-approximation algorithm for guarding 1.5-dimensional terrains. In *LATIN 2006: Theoretical Informatics, 7th Latin American Symposium, Valdivia, Chile, March 20-24, 2006, Proceedings*, pages 629–640, 2006. doi:10.1007/11682462_58.

17 James King and David G. Kirkpatrick. Improved approximation for guarding simple galleries from the perimeter. *Discrete & Computational Geometry*, 46(2):252–269, 2011. doi:10.1007/s00454-011-9352-x.

18 James King and Erik Krohn. Terrain guarding is NP-hard. *SIAM J. Comput.*, 40(5):1316–1339, 2011. doi:10.1137/100791506.

19 David G. Kirkpatrick. An o(lg lg opt)-approximation algorithm for multi-guarding galleries. *Discrete & Computational Geometry*, 53(2):327–343, 2015. doi:10.1007/s00454-014-9656-8.

20 Erik Krohn, Matt Gibson, Gaurav Kanade, and Kasturi R. Varadarajan. Guarding terrains via local search. *JoCG*, 5(1):168–178, 2014. URL: http://jocg.org/index.php/jocg/article/view/128.

21 Erik Krohn and Bengt J. Nilsson. Approximate guarding of monotone and rectilinear polygons. *Algorithmica*, 66(3):564–594, 2013. doi:10.1007/s00453-012-9653-3.

22 David Lichtenstein. Planar formulae and their uses. *SIAM J. Comput.*, 11(2):329–343, 1982. doi:10.1137/0211025.

23 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the exponential time hypothesis. *Bulletin of the EATCS*, 105:41–72, 2011. URL: http://albcom.lsi.upc.edu/ojs/index.php/beatcs/article/view/96.

24 Nabil H. Mustafa and Saurabh Ray. Improved results on geometric hitting set problems. *Discrete & Computational Geometry*, 44(4):883–895, 2010. doi:10.1007/s00454-010-9285-9.

QPTAS and Subexponential Algorithm for Maximum Clique on Disk Graphs

Édouard Bonnet[1]
ENS Lyon, LIP
Lyon, France
edouard.bonnet@dauphine.fr

Panos Giannopoulos[2]
Department of Computer Science, Middlesex University
London, UK
p.giannopoulos@mdx.ac.uk

Eun Jung Kim
Université Paris-Dauphine, PSL Research University, CNRS UMR, LAMSADE
Paris, France
eun-jung.kim@dauphine.fr

Paweł Rzążewski
Faculty of Mathematics and Information Science, Warsaw University of Technology
Warsaw, Poland
p.rzazewski@mini.pw.edu.pl

Florian Sikora[3]
Université Paris-Dauphine, PSL Research University, CNRS UMR, LAMSADE
Paris, France
florian.sikora@dauphine.fr

Abstract

A (unit) disk graph is the intersection graph of closed (unit) disks in the plane. Almost three decades ago, an elegant polynomial-time algorithm was found for Maximum Clique on unit disk graphs [Clark, Colbourn, Johnson; Discrete Mathematics '90]. Since then, it has been an intriguing open question whether or not tractability can be extended to general disk graphs. We show the rather surprising structural result that a disjoint union of cycles is the complement of a disk graph if and only if at most one of those cycles is of odd length. From that, we derive the first QPTAS and subexponential algorithm running in time $2^{\tilde{O}(n^{2/3})}$ for Maximum Clique on disk graphs. In stark contrast, Maximum Clique on intersection graphs of filled ellipses or filled triangles is unlikely to have such algorithms, even when the ellipses are close to unit disks. Indeed, we show that there is a constant ratio of approximation which cannot be attained even in time $2^{n^{1-\varepsilon}}$, unless the Exponential Time Hypothesis fails.

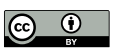
2012 ACM Subject Classification Theory of computation → Computational geometry

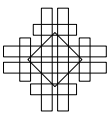
Keywords and phrases disk graph, maximum clique, computational complexity

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.12

Related Version A full version of this paper is available at `https://arxiv.org/abs/1712.05010`

[1] Supported by EPSRC grant FptGeom (EP/N029143/1)
[2] Partially supported by EPSRC grant FptGeom (EP/N029143/1)
[3] Partially supported by project ESIGMA (ANR-17-CE40-0028)





1 Introduction

An *intersection graph* of geometric objects has one vertex per object and an edge between every pair of vertices corresponding to intersecting objects. Intersection graphs for many different families of geometric objects have been studied due to their practical applications and rich structural properties [31, 10]. Among the most studied ones are *disk graphs*, which are intersection graphs of closed disks in the plane, and their special case, *unit disk graphs*, where all the radii are the same. Their applications range from sensor networks to map labeling [21], and many standard optimization problems have been studied on disk graphs, see for example [36] and references therein. In this paper, we study Maximum Clique on general disk graphs.

Known results

Recognizing unit disk graphs is NP-hard [11], and even $\exists\mathbb{R}$-complete [25]. Clark et al. [19] gave a polynomial-time algorithm for Maximum Clique on unit disk graphs with a geometric representation. The core idea of their algorithm can actually be adapted so that the geometric representation is no longer needed [34]. The complexity of the problem on general disk graphs is unfortunately still unknown. Using the fact that the transversal number for disks is 4, Ambühl and Wagner [4] gave a simple 2-approximation algorithm for Maximum Clique on general disk graphs. They also showed the problem to be APX-hard on intersection graphs of ellipses and gave a $9\rho^2$-approximation algorithm for filled ellipses of aspect ratio at most ρ. Since then, the problem has proved to be elusive with no new positive or negative results. The question on the complexity and further approximability of Maximum Clique on general disk graphs is considered as folklore [6], but was also explicitly mentioned as an open problem by Fishkin [21], Ambühl and Wagner [4] and Cabello [13, 14].

A closely related problem is Maximum Independent Set, which is W[1]-hard (even on unit disk graphs [30]), and admits a subexponential exact algorithm [2] and a PTAS [20, 17] on disk graphs.

Results and organization

In Section 2, we mainly prove that the disjoint union of two odd cycles is not the complement of a disk graph. To the best of our knowledge, this is the first structural property that general disk graphs do not inherit from strings or from convex objects. We provide an infinite family of forbidden induced subgraphs, an analogue to the recent work of Atminas and Zamaraev on unit disk graphs [5]. In Section 3, we show how to use this structural result to approximate and solve Maximum Independent Set on complements of disk graphs, hence Maximum Clique on disk graphs. More precisely, we present the first quasi-polynomial-time approximation scheme (QPTAS) and subexponential-time algorithm for Maximum Clique on disk graphs, even without the geometric representation of the graph. In Section 4, we highlight how those algorithms contrast with the situation for ellipses or triangles, where there is a constant $\alpha > 1$ for which an α-approximation running in subexponential time is highly unlikely (in particular, ruling out at once QPTAS *and* subexponential-time algorithm). We conclude in Section 5 with a few open questions.

Definitions and notations

For two integers $i \leqslant j$, we denote by $[i, j]$ the set of integers $\{i, i+1, \ldots, j-1, j\}$. For a positive integer i, we denote by $[i]$ the set of integers $[1, i]$. If S is a subset of vertices of a

graph, we denote by $N(S)$ the open neighborhood of S and by $N[S]$ the set $N(S) \cup S$. The *2-subdivision* of a graph G is the graph H obtained by subdividing each edge of G exactly twice. If G has n vertices and m edges, then H has $n + 2m$ vertices and $3m$ edges. The *co-2-subdivision* of G is the complement of H. Hence, it has $n+2m$ vertices and $\binom{n+2m}{2} - 3m$ edges. The *co-degree* of a graph is the maximum degree of its complement. A *co-disk* is a graph that is the complement of a disk graph.

For two distinct points x and y in the plane, we denote by $\ell(x, y)$ the unique line going through x and y, and by $\mathrm{seg}(x, y)$ the closed straight-line segment whose endpoints are x and y. If s is a segment with positive length, then we denote by $\ell(s)$ the unique line containing s. We denote by $d(x, y)$ the euclidean distance between points x and y. We will often define disks and elliptical disks by their boundary, i.e., circles and ellipses, and also use the following basic facts. There are exactly two circles that pass through a given point with a given tangent at this point and a given radius; one if we further specify on which side of the tangent the circle is. There is exactly one circle which passes through two points with a given tangent at one of the two points, provided the other point is *not* on this tangent. Finally, there exists one (not necessarily unique) ellipse which passes through two given points with two given tangents at those points.

The *Exponential Time Hypothesis* (ETH) is a conjecture by Impagliazzo et al. asserting that there is no $2^{o(n)}$-time algorithm for 3-SAT on instances with n variables [24]. The ETH, together with the sparsification lemma [24], even implies that there is no $2^{o(n+m)}$-time algorithm solving 3-SAT.

2 Disk graphs with co-degree 2

In this section, we fully characterize the degree-2 complements of disk graphs. We show the following:

▶ **Theorem 1.** *A disjoint union of paths and cycles is the complement of a disk graph if and only if the number of odd cycles is at most one.*

We split this theorem into two parts. In the first one, Section 2.1, we show that the union of two disjoint odd cycles is not the complement of a disk graph. This is the part that will be algorithmically useful. As disk graphs are closed under taking induced subgraphs, it implies that in the complement of a disk graph two vertex-disjoint odd cycles have to be linked by at least one edge. This will turn out useful when solving Maximum Independent Set on the complement of the graph (to solve Maximum Clique on the original graph). In the second part, Section 2.2, we show how to represent the complement of the disjoint union of even cycles and exactly one odd cycle. Although this result is not needed for the forthcoming algorithmic section, it nicely highlights the singular role that parity plays and exposes the complete set of disk graphs of co-degree 2.

2.1 The disjoint union of two odd cycles is not co-disk

We call *positive distance* between two non-intersecting disks the minimum of $d(x, y)$ where x is in one disk and y is in the other. If the disks are centered at c_1 and c_2 with radius r_1 and r_2, respectively, then this value is $d(c_1, c_2) - r_1 - r_2$. We call *negative distance* between two intersecting disks the length of the straight-line segment defined as the intersection of three objects: the two disks and the line joining their center. This value is $r_1 + r_2 - d(c_1, c_2)$, which is positive.

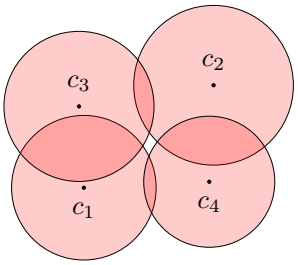


Figure 1 Disk realization of a $K_{2,2}$. As the centers are positioned, it is impossible that the two non-edges are between the disks 2 and 3, and between the disks 1 and 4 (or between the disks 1 and 3, and between the disks 2 and 4).

We call *proper representation* a disk representation where every edge is witnessed by a proper intersection of the two corresponding disks, i.e., the interiors of the two disks intersect. It is easy to transform a disk representation into a proper representation (of the same graph).

▶ **Lemma 2.** *If a graph has a disk representation, then it has a proper representation.*

Proof. If two disks intersect non-properly, we increase the radius of one of them by $\varepsilon/2$ where ε is the smallest positive distance between two disks. ◀

In order not to have to discuss about the corner case of three aligned centers in a disk representation, we show that such a configuration is never needed to represent a disk graph.

▶ **Lemma 3.** *If a graph has a disk representation, it has a proper representation where no three centers are aligned.*

Proof. By Lemma 2, we have or obtain a proper representation. Let ε be the minimum between the smallest positive distance and the smallest negative distance. As the representation is proper, $\varepsilon > 0$. If three centers are aligned, we move one of them to any point which is not lying in a line defined by two centers in a ball of radius $\varepsilon/2$ centered at it. This decreases by at least one the number of triple of aligned centers, and can be repeated until no three centers are aligned. ◀

From now on, we assume that every disk representation is proper and without three aligned centers. We show the folklore result that in a representation of a $K_{2,2}$ that sets the four centers in convex position, both non-edges have to be *diagonal*.

▶ **Lemma 4.** *In a disk representation of $K_{2,2}$ with the four centers in convex position, the non-edges are between vertices corresponding to opposite centers in the quadrangle.*

Proof. Let c_1 and c_2 be the centers of one non-edge, and c_3 and c_4 the centers of the other non-edge. Let r_i be the radius associated to center c_i for $i \in [4]$. It should be that $d(c_1, c_2) > r_1 + r_2$ and $d(c_3, c_4) > r_3 + r_4$ (see Figure 1). Assume c_1 and c_2 are consecutive on the convex hull formed by $\{c_1, c_2, c_3, c_4\}$, and say, without loss of generality, that the order is c_1, c_2, c_3, c_4. Let c be the intersection of $\text{seg}(c_1, c_3)$ and $\text{seg}(c_2, c_4)$. It holds that $d(c_1, c_3) + d(c_2, c_4) = d(c_1, c) + d(c, c_3) + d(c_2, c) + d(c, c_4) = (d(c_1, c) + d(c, c_2)) + (d(c_3, c) + d(c, c_4)) > d(c_1, c_2) + d(c_3, c_4) > r_1 + r_2 + r_3 + r_4 = (r_1 + r_3) + (r_2 + r_4)$. Which implies that $d(c_1, c_3) > r_1 + r_3$ or $d(c_2, c_4) > r_2 + r_4$; a contradiction. ◀

We derive a useful consequence of the previous lemma, phrased in terms of intersections of lines and segments.

▶ **Corollary 5.** *In any disk representation of $K_{2,2}$ with centers c_1, c_2, c_3, c_4 with the two non-edges between the vertices corresponding to c_1 and c_2, and between c_3 and c_4, it should be that $\ell(c_1, c_2)$ intersects $seg(c_3, c_4)$ or $\ell(c_3, c_4)$ intersects $seg(c_1, c_2)$.*

Proof. The disk representation either has the four centers in convex position or it has one center, say, without loss of generality, c_1, in the interior of the triangle formed by the other three centers. Then, in the first case, by Lemma 4, $\text{seg}(c_1, c_2)$ and $\text{seg}(c_3, c_4)$ are the diagonals of a convex quadrangle. Hence they intersect, and *a fortiori*, $\ell(c_1, c_2)$ intersects $\text{seg}(c_3, c_4)$ ($\ell(c_3, c_4)$ intersects $\text{seg}(c_1, c_2)$, too). In the second case, $\ell(c_1, c_2)$ intersects $\text{seg}(c_3, c_4)$. If instead a center in $\{c_3, c_4\}$ is in the interior of the triangle formed by the other centers, then $\ell(c_3, c_4)$ intersects $\text{seg}(c_1, c_2)$. ◀

We can now prove the main result of this section thanks to the previous corollary, parity arguments, and *some elementary properties of closed plane curves*, namely Property I and Property III of the eponymous paper [35].

▶ **Theorem 6.** *The complement of the disjoint union of two odd cycles is not a disk graph.*

Proof. Let s and t be two positive integers and $G = \overline{C_{2s+1} + C_{2t+1}}$ the complement of the disjoint union of a cycle of length $2s+1$ and a cycle of length $2t+1$. Assume that G is a disk graph. Let $\mathcal{C}_1$ (resp. $\mathcal{C}_2$) be the cycle embedded in the plane formed by $2s+1$ (resp. $2t+1$) straight-line segments joining the consecutive centers of disks along the first (resp. second) cycle. Observe that the segments of those two cycles correspond to the non-edges of G. We number the segments of $\mathcal{C}_1$ from S_1 to S_{2s+1}, and the segments of $\mathcal{C}_2$, from S'_1 to S'_{2t+1}.

For the i-th segment S_i of $\mathcal{C}_1$, let a_i be the number of segments of $\mathcal{C}_2$ intersected by the line $\ell(S_i)$ prolonging S_i, let b_i be the number of segments S'_j of $\mathcal{C}_2$ such that the prolonging line $\ell(S'_j)$ intersects S_i, and let c_i be the number of segments of $\mathcal{C}_2$ intersecting S_i. For the second cycle, we define similarly a'_j, b'_j, c'_j. The quantity $a_i + b_i - c_i$ counts the number of segments of $\mathcal{C}_2$ which can possibly represent a $K_{2,2}$ with S_i according to Corollary 5. As we assumed that G is a disk graph, $a_i + b_i - c_i = 2t + 1$ for every $i \in [2s + 1]$. Otherwise there would be at least one segment S'_j of $\mathcal{C}_2$ such that $\ell(S_i)$ does not intersect S'_j *and* $\ell(S'_j)$ does not intersect S_i.

Observe that a_i is an even integer since $\mathcal{C}_2$ is a closed curve. Also, $\Sigma_{i=1}^{2s+1} a_i + b_i - c_i = (2t + 1)(2s + 1)$ is an odd number, as the product of two odd numbers. This implies that $\Sigma_{i=1}^{2s+1} b_i - c_i$ shall be odd. $\Sigma_{i=1}^{2s+1} c_i$ counts the number of intersections of the two closed curves $\mathcal{C}_1$ and $\mathcal{C}_2$, and is therefore even. Hence, $\Sigma_{i=1}^{2s+1} b_i$ shall be odd. Observe that $\Sigma_{i=1}^{2s+1} b_i = \Sigma_{j=1}^{2t+1} a'_j$ by reordering and reinterpreting the sum from the point of view of the segments of $\mathcal{C}_2$. Since the a'_j are all even, $\Sigma_{i=1}^{2s+1} b_i$ is also even; a contradiction. ◀

2.2 The disjoint union of cycles with at most one odd is co-disk

We only show the following part of Theorem 1 to emphasize that, rather unexpectedly, parity plays a crucial role in disk graphs of co-degree 2. It is also amusing that the complement of any odd cycle is a *unit* disk graph while the complement of any even cycle of length at least 8 is not [5]. Here, the situation is somewhat reversed: complements of even cycles are *easier* to represent than complements of odd cycles. The proof of the following theorem can be found in the full version [9].

▶ **Theorem 7.** *The complement of the disjoint union of even cycles and one odd cycle is a disk graph.*

Theorem 6 and Theorem 7, together with the fact that disk graphs are closed by taking induced subgraphs prove Theorem 1.

3 Algorithmic consequences

Now we show how to use the structural results from Section 2 to obtain algorithms for Maximum Clique in disk graphs. A clique in a graph G is an independent set in $\overline{G}$. So, leveraging the result from Theorem 1, we will focus on solving Maximum Independent Set in graphs without two vertex-disjoint odd cycles as an induced subgraph.

3.1 QPTAS

The odd cycle packing number $\mathrm{ocp}(H)$ of a graph H is the maximum number of vertex-disjoint odd cycles in H. Unfortunately, the condition that $\overline{G}$ does not contain two vertex-disjoint odd cycles as an induced subgraph is not quite the same as saying that the odd cycle packing number of $\overline{G}$ is 1. Otherwise, we would immediately get a PTAS by the following result of Bock et al. [7].

▶ **Theorem 8** (Bock et al. [7])**.** *For every fixed $\varepsilon > 0$ there is a polynomial $(1+\varepsilon)$-approximation algorithm for* Maximum Independent Set *for graphs H with n vertices and $\mathrm{ocp}(H) = o(n/\log n)$.*

The algorithm by Bock et al. works in polynomial time if $\mathrm{ocp}(H) = o(n/\log n)$, but it does not need the odd cycle packing explicitly given as an input. This is important, since finding a maximum odd cycle packing is NP-hard [26]. We start by proving a structural lemma, which spares us having to determine the odd cycle packing number.

▶ **Lemma 9.** *Let H be a graph with n vertices, whose complement is a disk graph. If $\mathrm{ocp}(H) > n/\log^2 n$, then H has a vertex of degree at least $n/\log^4 n$.*

Proof. Consider a maximum odd cycle packing $\mathcal{C}$. By assumption, it contains more than $n/\log^2 n$ vertex-disjoint cycles. By the pigeonhole principle, there must be a cycle $C \in \mathcal{C}$ of size at most $\log^2 n$. Now, by Theorem 6, H has no two vertex-disjoint odd cycles with no edges between them. Therefore there must be an edge from C to every other cycle of $\mathcal{C}$, there are at least $n/\log^2 n$ such edges. Let v be a vertex of C with the maximum number of edges to other cycles in $\mathcal{C}$, by the pigeonhole principle its degree is at least $n/\log^4 n$. ◀

Now we are ready to construct a QPTAS for Maximum Clique in disk graphs.

▶ **Theorem 10.** *For any $\varepsilon > 0$,* Maximum Clique *can be $(1+\varepsilon)$-approximated in time $2^{O(\log^5 n)}$, when the input is a disk graph with n vertices.*

Proof. Let G be the input disk graph and let $\overline{G}$ be its complement, we want to find a $(1+\varepsilon)$-approximation for Maximum Independent Set in $\overline{G}$. We consider two cases. If $\overline{G}$ has no vertex of degree at least $n/\log^4 n$, then, by Lemma 9, we know that $\mathrm{ocp}(\overline{G}) \leqslant n/\log^2 n = o(n/\log n)$. In this case we run the PTAS of Bock et al. and we are done.

In the other case, $\overline{G}$ has a vertex v of degree at least $n/\log^4 n$ (note that it may still be the case that $\mathrm{ocp}(\overline{G}) = o(n/\log n)$). We branch on v: either we include v in our solution and remove it and all its neighbors, or we discard v. The complexity of this step is described by the recursion $F(n) \leqslant F(n-1) + F(n - n/\log^4 n)$ and solving it gives us the desired running time. Note that this step is exact, i.e., we do not lose any solutions. ◀

3.2 Subexponential algorithm

Now we will show how our structural result can be used to construct a subexponential algorithm for Maximum Clique in disk graphs. The *odd girth* of a graph is the size of a shortest odd cycle. An *odd cycle cover* is a subset of vertices whose deletion makes the graph bipartite. We will use a result by Györi et al. [23], which says that graphs with large odd girth have small odd cycle cover. In that sense, it can be seen as relativizing the fact that odd cycles do not have the Erdős-Pósa property. Bock et al. [7] turned the non-constructive proof into a polynomial-time algorithm.

▶ **Theorem 11** (Györi et al. [23], Bock et al. [7]). *Let H be a graph with n vertices and no odd cycle shorter than δn (δ may be a function of n). Then there is an odd cycle cover X of size at most $(48/\delta)\ln(5/\delta)$ Moreover, X can be found in polynomial time.*

Let us start with showing three variants of an algorithm.

▶ **Theorem 12.** *Let G be a disk graph with n vertices. Let Δ be the maximum degree of $\overline{G}$ and c the odd girth of $\overline{G}$ (they may be functions of n).* Maximum Clique *has a branching or can be solved, up to a polynomial factor, in time:*

(i) $2^{\tilde{O}(n/\Delta)}$ *(branching),*
(ii) $2^{\tilde{O}(n/c)}$ *(solved),*
(iii) $2^{O(c\Delta)}$ *(solved).*

Proof. Let G be the input disk graph and let $\overline{G}$ be its complement, we look for a maximum independent set in $\overline{G}$.

To prove i, consider a vertex v of degree Δ in $\overline{G}$. We branch on v: either we include v in our solution and remove $N[v]$, or discard v. The complexity is described by the recursion $F(n) \leqslant F(n-1) + F(n-(\Delta+1))$ and solving it gives i. Observe that this does not give an algorithm running in time $2^{\tilde{O}(n/\Delta)}$ since the maximum degree might drop. Therefore, we will do this branching as long as it is *good enough* and then finish with the algorithms corresponding to ii and iii.

For ii and iii, let C be a cycle of length c, it can be found in polynomial time (see for instance [3]). By application of Theorem 11 with $\delta = c/n$, we find an odd cycle cover X in $\overline{G}$ of size $\tilde{O}(n/c)$ in polynomial time. Next we exhaustively guess in time $2^{\tilde{O}(n/c)}$ the intersection I of an optimum solution with X and finish by finding a maximum independent set in the bipartite graph $\overline{G} - (X \cup N(I))$, which can be done in polynomial time. The total complexity of this case is $2^{\tilde{O}(n/c)}$, which shows ii.

Finally, observe that the graph $\overline{G} - N[C]$ is bipartite, since otherwise $\overline{G}$ contains two vertex-disjoint odd cycles with no edges between them. Moreover, since every vertex in $\overline{G}$ has degree at most Δ, it holds that $|N[C]| \leqslant c(\Delta-1) \leqslant c\Delta$. Indeed, a vertex of C can only have $c(\Delta-2)$ neighbors outside C. We can proceed as in the previous step: we exhaustively guess the intersection of the optimal solution with $N[C]$ and finish by finding the maximum independent set in a bipartite graph (a subgraph of $\overline{G} - N[C]$), which can be done in total time $2^{O(c\Delta)}$, which shows iii. ◀

Now we show how the structure of G affect the bounds in Theorem 12.

▶ **Corollary 13.** *Let G be a disk graph with n vertices.* Maximum Clique *can be solved in time:*

(a) $2^{\tilde{O}(n^{2/3})}$,
(b) $2^{\tilde{O}(\sqrt{n})}$ *if the maximum degree of $\overline{G}$ is constant,*
(c) *polynomial, if both the maximum degree and the odd girth of $\overline{G}$ are constant.*

Proof. Δ and c can be computed in polynomial time. Therefore, knowing what is faster among cases i, ii, and iii is tractable. For case (a), while there is a vertex of degree at least $n^{1/3}$, we branch on it. When this process stops, we do what is more advantageous between cases ii and iii. Note that $\min(n/\Delta, n/c, c\Delta) \leqslant n^{2/3}$ (the equality is met for $\Delta = c = n^{1/3}$). For case (b), we do what is best between cases ii and iii. Note that $\min(n/c, c) \leqslant \sqrt{n}$ (the equality is met for $c = \sqrt{n}$). Finally, case (c) follows directly from case iii in Theorem 12. ◀

Observe that case (b) is typically the hardest one for Maximum Clique. Moreover, the win-win strategy of Corollary 13 can be directly applied to solve Maximum Weighted Clique, as finding a maximum weighted independent set in a bipartite graph is still polynomial-time solvable. On the other hand, this approach cannot be easily adapted to obtain a subexponential algorithm for Clique Partition (even Clique p-Partition with constant p), since List Coloring (even List 3-Coloring) has no subexponential algorithm for bipartite graphs, unless the ETH fails (see [29], the bound can be obtained if we start reduction from a sparse instance of 1-in-3-Sat instead of Planar 1-in-3-Sat).

4 Other intersection graphs and limits

In this section, we discuss the impossibility of generalizing our results to related classes of intersection graphs.

4.1 Filled ellipses and filled triangles

A natural generalization of a disk is an *elliptical disk*, also called *filled ellipse*, i.e., an ellipse plus its interior. The simplest convex set with non empty interior is a filled triangle (a triangle plus its interior). We show that our approach developed in the two previous sections, and actually every approach, is bound to fail for filled ellipses and filled triangles.

APX-hardness was shown for Maximum Clique in the intersection graphs of (*non-filled*) ellipses and triangles by Ambühl and Wagner [4]. Their reduction also implies that there is no subexponential algorithm for this problem, unless the ETH fails. Moreover, they claim that their hardness result extends to filled ellipses since *"intersection graphs of ellipses without interior are also intersection graphs of filled ellipses"*. Unfortunately, this claim is incorrect. In the full version [9], we show:

▶ **Theorem 14.** *There is a graph G which has an intersection representation with ellipses without their interior, but has no intersection representation with convex sets.*

This error and the confusion between filled ellipses and ellipses without their interior has propagated to other more recent papers [27]. Fortunately, we show that the hardness result does hold for filled ellipses (and filled triangles) with a different reduction. Our construction can be seen as streamlining the ideas of Ambühl and Wagner [4]. It is simpler and, in the case of (filled) ellipses, yields a somewhat stronger statement.

▶ **Theorem 15.** *There is a constant $\alpha > 1$ such that for every $\varepsilon > 0$,* Maximum Clique *on the intersection graphs of filled ellipses has no α-approximation algorithm running in subexponential time $2^{n^{1-\varepsilon}}$, unless the ETH fails, even when the ellipses have arbitrarily small eccentricity and arbitrarily close value of major axis.*

This is in sharp contrast with our subexponential algorithm and with our QPTAS when the eccentricity is 0 (case of disks). For any $\varepsilon > 0$, if the eccentricity is only allowed to be at most ε, a subexponential algorithm or a QPTAS are very unlikely. This result subsumes

[16] (where NP-hardness is shown for connected shapes contained in a disk of radius 1 and containing a concentric disk of radius $1-\varepsilon$ for arbitrarily small $\varepsilon > 0$) and corrects [4]. We show the same hardness for the intersection graphs of filled triangles.

▶ **Theorem 16.** *There is a constant $\alpha > 1$ such that for every $\varepsilon > 0$,* Maximum Clique *on the intersection graphs of filled triangles has no α-approximation algorithm running in subexponential time $2^{n^{1-\varepsilon}}$, unless the ETH fails.*

We first show this lower bound for Maximum Weighted Independent Set on the class of all the 2-subdivisions, hence the same hardness for Maximum Weighted Clique on all the co-2-subdivisions. It is folklore that from the PCP of Moshkovitz and Raz [33], which roughly implies that Max 3-SAT cannot be $7/8+\varepsilon$-approximated in subexponential time under the ETH, one can derive such inapproximability in subexponential time for many hard graph and hypergraph problems; see for instance [8].

The following inapproximability result for Maximum Independent Set on bounded-degree graphs was shown by Chlebík and Chlebíková [18]. As their reduction is almost linear, the PCP of Moshkovitz and Raz boosts this hardness result from ruling out polynomial-time up to ruling out subexponential time $2^{n^{1-\varepsilon}}$ for any $\varepsilon > 0$.

▶ **Theorem 17** ([18, 33])**.** *There is a constant $\beta > 0$ such that* Maximum Independent Set *on graphs with n vertices and maximum degree Δ cannot be $1+\beta$-approximated in time $2^{n^{1-\varepsilon}}$ for any $\varepsilon > 0$, unless the ETH fails.*

We could actually state a slightly stronger statement for the running time but will settle for this for the sake of clarity.

▶ **Theorem 18.** *There is a constant $\alpha > 1$ such that for any $\varepsilon > 0$,* Maximum Independent Set *on the class of all the 2-subdivisions has no α-approximation algorithm running in subexponential time $2^{n^{1-\varepsilon}}$, unless the ETH fails.*

Proof. Let G be a graph with maximum degree a constant Δ, with n vertices $v_1, \ldots, v_n$ and m edges $e_1, \ldots, e_m$, and let H be its 2-subdivision. Recall that to form H, we subdivided every edge of G exactly twice. These $2m$ vertices in $V(H) \setminus V(G)$, representing edges, are called *edge vertices* and are denoted by $v^+(e_1), v^-(e_1), \ldots, v^+(e_m), v^-(e_m)$, as opposed to the other vertices of H, which we call *original vertices*. If $e_k = v_iv_j$ is an edge of G, then $v^+(e_k)$ (resp. $v^-(e_k)$) has two neighbors: $v^-(e_k)$ and v_i (resp. $v^+(e_k)$ and v_j).

Observe that there is a maximum independent set S which contains exactly one of $v^+(e_k), v^-(e_k)$ for every $k \in [m]$. Indeed, S cannot contain both $v^+(e_k)$ and $v^-(e_k)$ since they are adjacent. On the other hand, if S contains neither $v^+(e_k)$ nor $v^-(e_k)$, then adding $v^+(e_k)$ to S and potentially removing the other neighbor of $v^+(e_k)$ which is v_i (with $e_k = v_iv_j$) can only increase the size of the independent set. Hence S contains m edge vertices and $s \leqslant n$ original vertices, and there is no larger independent set in H.

We observe that the s original vertices is S form an independent set in G. Indeed, if $v_iv_j = e_k \in E(G)$ and $v_i, v_j \in S$, then neither $v^+(e_k)$ nor $v^-(e_k)$ could be in S.

Now, assume there is an approximation with ratio $\alpha := 1 + \frac{2\beta}{(\Delta+1)^2}$ for Maximum Independent Set on 2-subdivisions running in subexponential time, where $1+\beta > 1$ is a ratio which is not attainable for Maximum Independent Set on graphs of maximum degree Δ according to Theorem 17. On instance H, this algorithm would output a solution with m' edge vertices and s' original vertices. As we already observed this solution can be easily (in polynomial time) transformed into an at-least-as-good solution with m edge vertices and s'' original vertices forming an independent set in G. Further, we may assume

that $s'' \geqslant n/(\Delta+1)$ since for any independent set of G, we can obtain an independent set of H consisting of the same set of original vertices and m edge vertices. Since $m \leqslant n\Delta/2$ and $s'' \geqslant n/(\Delta+1)$, we obtain $m \leqslant s''\Delta(\Delta+1)/2$ and $2m/(\Delta+1)^2 \leqslant s''\Delta/(\Delta+1)$. From $\frac{m+s}{m+s''} \leqslant \alpha$ and $\Delta \geqslant 3$, we have

$$s \leqslant m \cdot \frac{2\beta}{(\Delta+1)^2} + s'' \cdot (1 + \frac{2\beta}{(\Delta+1)^2}) \leqslant s''(\frac{\Delta\beta}{\Delta+1} + 1 + \frac{2\beta}{(\Delta+1)^2}) \leqslant s''(1+\beta)$$

This contradicts the inapproximability of Theorem 17. Indeed, note that the number of vertices of H is only a constant times the number of vertices of G (recall that G has bounded maximum degree, hence $m = O(n)$). ◀

Recalling that independent set is a clique in the complement, we get the following.

▶ **Corollary 19.** *There is a constant $\alpha > 1$ such that for any $\varepsilon > 0$,* Maximum Clique *on the class of all the co-2-subdivisions has no α-approximation algorithm running in subexponential time $2^{n^{1-\varepsilon}}$, unless the ETH fails.*

For exact algorithms the subexponential time that we rule out under the ETH is not only $2^{n^{1-\varepsilon}}$ but actually any $2^{o(n)}$.

Now, to Theorem 15 and Theorem 16, it is sufficient to show that intersection graphs of (filled) ellipses or of (filled) triangles contain all co-2-subdivisions. We start with (filled) triangles since the construction is straightforward.

▶ **Lemma 20.** *The class of intersection graphs of filled triangles contains all co-2-subdivisions.*

Proof. Let G be any graph with n vertices $v_1, \ldots, v_n$ and m edges $e_1, \ldots, e_m$, and H be its co-2-subdivision. We start with $n+2$ points $p_0, p_1, p_2, \ldots, p_n, p_{n+1}$ forming a convex monotone chain. Those points can be chosen as $p_i := (i, p(i))$ where p is the equation of a positive parabola taking its minimum at $(0,0)$. For each $i \in [0, n+1]$, let q_i be the reflection of p_i by the line of equation $y = 0$. Let $x := (n+1, 0)$. For each vertex $v_i \in V(G)$ the filled triangle $\delta_i := p_i q_i x$ encodes v_i. Observe that the points $p_0 = q_0$, p_{n+1}, and q_{n+1} will only be used to define the filled triangles encoding edges.

To encode (the two new vertices of) a subdivided edge $e_k = v_i v_j$, we use two filled triangles Δ_k^+ and Δ_k^-. The triangle Δ_k^+ (resp. Δ_k^-) has an edge which is supported by $\ell(p_{i-1}, p_{i+1})$ (resp. $\ell(q_{j-1}, q_{j+1})$) and is prolonged so that it crosses the boundary of each $\delta_{i'}$ but δ_i (resp. but δ_j). A second edge of Δ_k^+ and Δ_k^- are parallel and make with the horizontal a small angle εk, where $\varepsilon > 0$ is chosen so that εm is smaller than the angle formed by $\ell(p_0, p_1)$ with the horizontal line. Those almost horizontal edges intersect for each pair $\Delta_{k'}^+$ and $\Delta_{k''}^-$ with $k' \neq k''$ intersects close to the same point. Filled triangles Δ_k^+ and Δ_k^- do not intersect. See Figure 2 for the complete picture.

It is easy to check that the intersection graph of $\{\delta_i\}_{i\in[n]} \cup \{\Delta_k^+, \Delta_k^-\}_{k\in[m]}$ is H. The family $\{\delta_i\}_{i\in[n]}$ forms a clique since they all contain for instance the point x. The filled triangle Δ_k^+ (resp. Δ_k^-) intersects every other filled triangles except Δ_k^- (resp. Δ_k^+) and δ_i (resp. δ_j) with $e_k = v_i v_j$.

One may observe that no triangle is fully included in another triangle. So the construction works both as the intersection graph of filled triangles *and* triangles without their interior. The edge of a Δ_k^+ or a Δ_k^- crossing the boundary of all but one δ_i, and the almost horizontal edge can be arbitrary prolonged to the right and to the left respectively. Thus, the triangles can all be made isosceles. ◀

We use the same ideas for the construction with filled ellipses. The two important sides of a triangle encoding an edge of the initial graph G become two tangents of the ellipse.

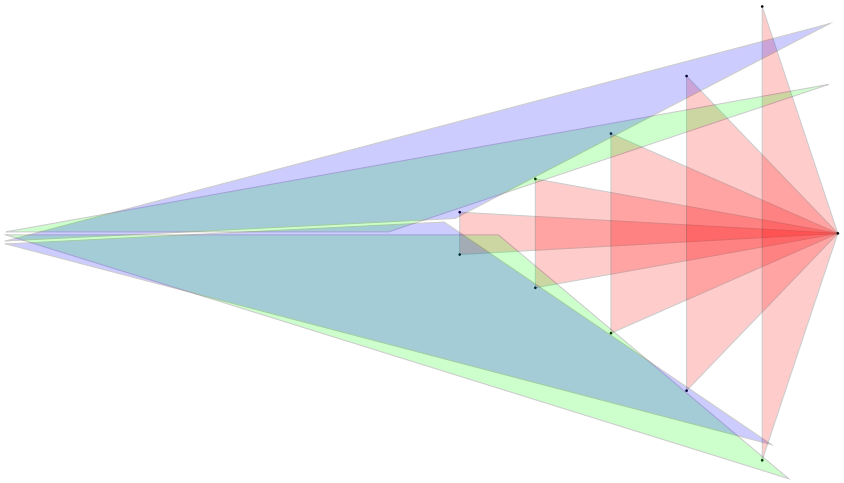


Figure 2 A co-2-subdivision of a graph with 5 vertices (in red) represented with triangles. Only two edges are shown: one between vertices 1 and 4 (green) and one between vertices 2 and 3 (blue).

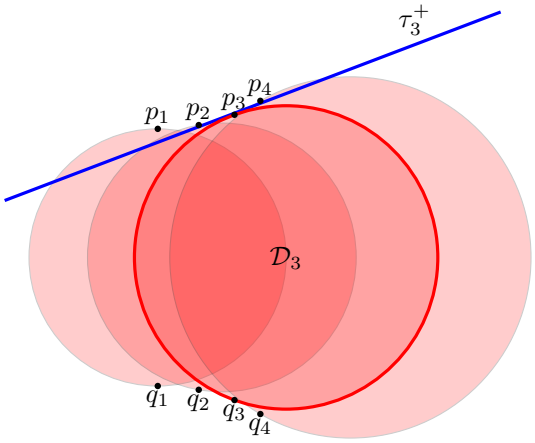


Figure 3 The blue line intersects every red disk but the third one.

▶ **Lemma 21.** *The class of intersection graphs of filled ellipses contains all co-2-subdivisions.*

Proof. Let G be any graph with n vertices $v_1, \ldots, v_n$ and m edges $e_1, \ldots, e_m$, and H be its co-2-subdivision. We start with the convex monotone chain $p_0, p_1, p_2, \ldots, p_{n-1}, p_n, p_{n+1}$, only the gap between p_i and p_{i+1} is chosen very small compared to the positive y-coordinate of p_0. The disks $\mathcal{D}_i$ encoding the vertices $v_i \in G$ must form a clique. We also take p_0 with a large x-coordinate. For $i \in [0, n+1]$, q_i is the symmetric of p_i with respect to the x-axis. For each $i \in [n]$, we define $\mathcal{D}_i$ as the disk whose boundary is the unique circle which goes through p_i and q_i, and whose tangent at p_i has the direction of $\ell(p_{i-1}, p_{i+1})$. It can be observed that, by symmetry, the tangent of $\mathcal{D}_i$ at q_i has the direction of $\ell(q_{i-1}, q_{i+1})$.

Let us call τ_i^+ (resp. τ_i^-) the tangent of $\mathcal{D}_i$ at p_i (resp. at q_i) very slightly translated upward (resp. downward). The tangent τ_i^+ (resp. τ_i^-) intersects every disks $\mathcal{D}_{i'}$ but $\mathcal{D}_i$ (see Figure 3). Let denote by p_i' (resp. q_i') be the projection of p_i (resp. q_i) onto τ_i^+ (resp. onto τ_i^-) For each $k \in [m]$, let ℓ_k be the line crossing the origin $O = (0, 0)$ and forming with the horizontal an angle εk, where εk is smaller than the angle formed by $\ell(p_0, p_1)$ with the horizontal. Let ℓ_k^+ (resp. ℓ_k^-) be ℓ_k very slightly translated upward (resp. downward). To

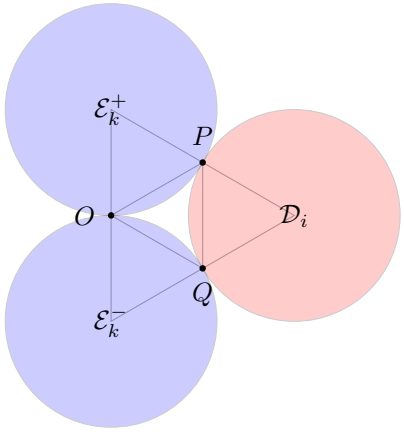


Figure 4 The layout of the disks $\mathcal{D}_i$, and the elliptical disks $\mathcal{E}_k^+$ and $\mathcal{E}_k^-$.

encode an edge $e_k = v_iv_j$, we have two filled ellipses $\mathcal{E}_k^+$ and $\mathcal{E}_k^-$. The ellipse $\mathcal{E}_k^+$ (resp. $\mathcal{E}_k^-$) is defined as being tangent with τ_i^+ at p_i' (resp. with τ_j^- at q_j') and tangent at ℓ_k^+ (resp. ℓ_k^-) at the point of x-coordinate 0 (thus very close to O), where $e_k = v_iv_j$. The proof that the intersection graph of $\{\mathcal{D}_i\}_{i\in[n]} \cup \{\mathcal{E}_k^+, \mathcal{E}_k^-\}_{k\in[m]}$ is H is similar to the case of filled triangles.

As no ellipse is fully contained in another ellipse, this construction works for both filled ellipses *and* ellipses without their interior.

We place p_0 at $P := (\sqrt{3}/2, 1/2)$ and make the distance between p_i and p_{i+1} very small compared to 1. All points p_i are very close to P and all points q_i are very close to $Q := (\sqrt{3}/2, -1/2)$. This makes the radius of all disks $\mathcal{D}_i$ arbitrarily close to 1. We choose the convex monotone chain $p_0, \ldots, p_{n+1}$ so that $\ell(p_0, p_1)$ forms a 60-degree angle with the horizontal. As, the chain is strictly convex but very close to a straight-line, $\ell(p_0, p_1) \approx \ell(p_n, p_{n+1}) \approx \ell(p_i, p_{i+1}) \approx \ell(p_i, p_{i+2})$. Thus, all those lines almost cross P and form an angle of roughly 60-degree with the horizontal. The same holds for points q_i. For the choice of an elliptical disk tangent to the x-axis at O and to a line with a 60-degree slope at P (resp. at Q), we take a disk of radius 1 centered at $(0, 1)$ (resp. at $(0, -1)$); see Figure 4.

The acute angle formed by ℓ_1 and ℓ_m (incident in O) is made arbitrarily small so that, by continuity of the elliptical disk defined by two tangents at two points, the filled ellipses $\mathcal{E}_k^+$ and $\mathcal{E}_k^-$ have eccentricity arbitrarily close to 0 and major axis arbitrarily close to 1. ◀

In the construction, we made *both* the eccentricity of the (filled) ellipses arbitrarily close to 0 and the ratio between the largest and the smallest major axis arbitrarily close to 1. We know that this construction is very unlikely to work for the extreme case of unit disks, since a polynomial algorithm is known for Max Clique. Note that even with disks of arbitrary radii, Theorem 6 unconditionally proves that the construction does fail. Indeed the co-2-subdivision of $C_3 + C_3$ is the complement of $C_9 + C_9$, hence not a disk graph.

4.2 Homothets of a convex polygon

Another natural direction of generalizing a result on disk intersection graphs is to consider *pseudodisk intersection graphs*, i.e., intersection graphs of collections of closed subsets of the plane (regions bounded by simple Jordan curves) that are pairwise in a *pseudodisk* relationship (see Kratochvíl [28]). Two regions A and B are in pseudodisk relation if both differences $A \setminus B$ and $B \setminus A$ are arc-connected. It is known that P_{hom} graphs, i.e., intersection graphs of homothetic copies of a fixed polygon P, are pseudodisk intersection graphs [1]. As

shown by Brimkov *et al.*, for every convex k-gon P, a P_{hom} graph with n vertices has at most n^k maximal cliques [12]. This clearly implies that Maximum Clique, but also Clique p-Partition for fixed p is polynomially solvable in P_{hom} graphs. Actually, the bound on the maximum number of maximal cliques from [12] holds for a more general class of graphs, called k_{DIR}-CONV, which admit a intersection representation by convex polygons, whose every side is parallel to one of k directions.

Moreover, we observe that Theorem 7 cannot be generalized to P_{hom} graphs or k_{DIR}-CONV graphs. Indeed, consider the complement $\overline{P_n}$ of an n-vertex path P_n. The number of maximal cliques in $\overline{P_n}$, or, equivalently, maximal independent sets in P_n is $\Theta(c^n)$ for $c \approx 1.32$, i.e., exponential in n [22]. Therefore, for every fixed polygon P (or for every fixed k) there is n, such that $\overline{P_n}$ is not a P_{hom} (k_{DIR}-CONV) graph.

5 Perspectives

We presented the first QPTAS and subexponential algorithm for Maximum Clique on disk graphs. Our subexponential algorithm extends to the weighted case and yields a polynomial algorithm if both the degree Δ and the odd girth c of the complement graph are constant. Indeed, our full characterization of disk graphs with co-degree 2, implies a backdoor-to-bipartiteness of size $c\Delta$ in the complement.

We have also paved the way for a potential NP-hardness construction. We showed why the versatile approach of representing complements of even subdivisions of graphs forming a class on which Maximum Independent Set is NP-hard fails if the class is *general graphs*, *planar graphs*, or even any class containing the disjoint union of two odd cycles. This approach was used by Middendorf for some string graphs [32] (with the class of all graphs), Cabello et al. [15] to settle the then long-standing open question of the complexity of Maximum Clique for segments (with the class of planar graphs), in Section 4 of this paper for ellipses and triangles (with the class of all graphs). Determining the complexity of Maximum Independent Set on graphs without two vertex-disjoint odd cycles as an induced subgraph is a valuable first step towards settling the complexity of Maximum Clique on disks.

Another direction is to try and strengthen our QPTAS in one of two ways: either to obtain a PTAS for Maximum Clique on disk graphs, or to obtain a QPTAS (or PTAS) for Maximum Weighted Clique on disk graphs. It is interesting to note that Bock et al. [7] showed a PTAS for Maximum Weighted Independent Set for graphs G with $\mathrm{ocp}(G) = O(\log n / \log\log n)$. However, this bound is too weak to use a win-win approach similar to Theorem 10.

References

1. Pankaj K Agarwal, János Pach, and Micha Sharir. State of the Union (of Geometric Objects). *Surveys in Discrete and Computational Geometry: Twenty Years Later. Contemporary Mathematics*, 453:9–48, 2008.
2. Jochen Alber and Jirí Fiala. Geometric separation and exact solutions for the parameterized independent set problem on disk graphs. *J. Algorithms*, 52(2):134–151, 2004. doi:10.1016/j.jalgor.2003.10.001.
3. Noga Alon, Raphael Yuster, and Uri Zwick. Finding and counting given length cycles. *Algorithmica*, 17(3):209–223, 1997. doi:10.1007/BF02523189.
4. Christoph Ambühl and Uli Wagner. The Clique Problem in Intersection Graphs of Ellipses and Triangles. *Theory Comput. Syst.*, 38(3):279–292, 2005. doi:10.1007/s00224-005-1141-6.

5 Aistis Atminas and Viktor Zamaraev. On forbidden induced subgraphs for unit disk graphs. *arXiv preprint arXiv:1602.08148*, 2016.

6 Jørgen Bang-Jensen, Bruce Reed, Mathias Schacht, Robert Šámal, Bjarne Toft, and Uli Wagner. On six problems posed by Jarik Nešetřil. *Topics in Discrete Mathematics*, pages 613–627, 2006.

7 Adrian Bock, Yuri Faenza, Carsten Moldenhauer, and Andres J. Ruiz-Vargas. Solving the Stable Set Problem in Terms of the Odd Cycle Packing Number. In *34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2014, December 15-17, 2014, New Delhi, India*, pages 187–198, 2014. doi:10.4230/LIPIcs.FSTTCS.2014.187.

8 Édouard Bonnet, Bruno Escoffier, Eun Jung Kim, and Vangelis Th. Paschos. On Subexponential and FPT-Time Inapproximability. *Algorithmica*, 71(3):541–565, 2015. doi:10.1007/s00453-014-9889-1.

9 Édouard Bonnet, Panos Giannopoulos, Eun Jung Kim, Paweł Rzążewski, and Florian Sikora. QPTAS and subexponential algorithm for maximum clique on disk graphs. *CoRR*, abs/1712.05010, 2017. arXiv:1712.05010.

10 Andreas Brandstädt, Van Bang Le, and Jeremy P Spinrad. *Graph classes: a survey*. SIAM, 1999.

11 Heinz Breu and David G. Kirkpatrick. Unit disk graph recognition is NP-hard. *Comput. Geom.*, 9(1-2):3–24, 1998. doi:10.1016/S0925-7721(97)00014-X.

12 Valentin E. Brimkov, Konstanty Junosza-Szaniawski, Sean Kafer, Jan Kratochvíl, Martin Pergel, Paweł Rzążewski, Matthew Szczepankiewicz, and Joshua Terhaar. Homothetic polygons and beyond: Intersection graphs, recognition, and maximum clique. *CoRR*, abs/1411.2928, 2014. arXiv:1411.2928.

13 Sergio Cabello. Maximum clique for disks of two sizes. Open problems from Geometric Intersection Graphs: Problems and Directions CG Week Workshop, Eindhoven, June 25, 2015 (http://cgweek15.tcs.uj.edu.pl/problems.pdf), 2015. [Online; accessed 07-December-2017].

14 Sergio Cabello. Open problems presented at the Algorithmic Graph Theory on the Adriatic Coast workshop, Koper, Slovenia (https://conferences.matheo.si/event/6/picture/35.pdf), June 16-19 2015.

15 Sergio Cabello, Jean Cardinal, and Stefan Langerman. The clique problem in ray intersection graphs. *Discrete & Computational Geometry*, 50(3):771–783, 2013. doi:10.1007/s00454-013-9538-5.

16 Stephan Ceroi. The clique number of unit quasi-disk graphs. Technical Report RR-4419, INRIA, 2002. URL: https://hal.inria.fr/inria-00072169.

17 Timothy M. Chan. Polynomial-time approximation schemes for packing and piercing fat objects. *J. Algorithms*, 46(2):178–189, 2003. doi:10.1016/S0196-6774(02)00294-8.

18 Miroslav Chlebík and Janka Chlebíková. Complexity of approximating bounded variants of optimization problems. *Theor. Comput. Sci.*, 354(3):320–338, 2006. doi:10.1016/j.tcs.2005.11.029.

19 Brent N. Clark, Charles J. Colbourn, and David S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86(1-3):165–177, 1990. doi:10.1016/0012-365X(90)90358-O.

20 Thomas Erlebach, Klaus Jansen, and Eike Seidel. Polynomial-time approximation schemes for geometric intersection graphs. *SIAM J. Comput.*, 34(6):1302–1323, 2005. doi:10.1137/S0097539702402676.

21 Aleksei V. Fishkin. Disk graphs: A short survey. In Klaus Jansen and Roberto Solis-Oba, editors, *Approximation and Online Algorithms, First International Workshop, WAOA 2003, Budapest, Hungary, September 16-18, 2003, Revised Papers*, volume 2909 of *Lecture Notes*

in Computer Science, pages 260–264. Springer, 2003. doi:10.1007/978-3-540-24592-6_23.

22 Zoltán Füredi. The number of maximal independent sets in connected graphs. *Journal of Graph Theory*, 11(4):463–470, 1987. doi:10.1002/jgt.3190110403.

23 Ervin Györi, Alexandr V Kostochka, and Tomasz Łuczak. Graphs without short odd cycles are nearly bipartite. *Discrete Mathematics*, 163(1):279–284, 1997. doi:10.1016/0012-365X(95)00321-M.

24 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

25 Ross J. Kang and Tobias Müller. Sphere and Dot Product Representations of Graphs. *Discrete & Computational Geometry*, 47(3):548–568, 2012. doi:10.1007/s00454-012-9394-8.

26 Ken-ichi Kawarabayashi and Bruce A. Reed. Odd cycle packing. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 695–704, 2010. doi:10.1145/1806689.1806785.

27 Chaya Keller, Shakhar Smorodinsky, and Gábor Tardos. On Max-Clique for intersection graphs of sets and the Hadwiger-Debrunner numbers. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2254–2263, 2017. doi:10.1137/1.9781611974782.148.

28 Jan Kratochvíl. Intersection graphs of noncrossing arc-connected sets in the plane. In *Graph Drawing, Symposium on Graph Drawing, GD '96, Berkeley, California, USA, September 18-20, Proceedings*, pages 257–270, 1996. doi:10.1007/3-540-62495-3_53.

29 J. Kratochvíl. Precoloring extension with fixed color bound. *Acta Mathematica Universitatis Comenianae. New Series*, 62(2):139–153, 1993. URL: http://eudml.org/doc/118661.

30 Dániel Marx. Parameterized Complexity and Approximation Algorithms. *Comput. J.*, 51(1):60–78, 2008. doi:10.1093/comjnl/bxm048.

31 Terry A McKee and Fred R McMorris. *Topics in intersection graph theory*. SIAM, 1999.

32 Matthias Middendorf and Frank Pfeiffer. The max clique problem in classes of string-graphs. *Discrete Mathematics*, 108(1-3):365–372, 1992. doi:10.1016/0012-365X(92)90688-C.

33 Dana Moshkovitz and Ran Raz. Two-query PCP with subconstant error. *J. ACM*, 57(5):29:1–29:29, 2010. doi:10.1145/1754399.1754402.

34 Vijay Raghavan and Jeremy P. Spinrad. Robust algorithms for restricted domains. *J. Algorithms*, 48(1):160–172, 2003. doi:10.1016/S0196-6774(03)00048-8.

35 Peter Guthrie Tait. Some elementary properties of closed plane curves. *Messenger of Mathematics, New Series*, 69:270–272, 1877.

36 Erik Jan van Leeuwen. *Optimization and Approximation on Systems of Geometric Objects*. PhD thesis, Utrecht University, 2009.

Computational Complexity of the Interleaving Distance

Håvard Bakke Bjerkevik
Department of Mathematical Sciences, Norwegian University of Science and Technology
Trondheim, Norway
havard.bjerkevik@ntnu.no

Magnus Bakke Botnan[1]
Department of Mathematics, TU Munich
Garching bei München, Germany
botnan@ma.tum.de

Abstract

The interleaving distance is arguably the most prominent distance measure in topological data analysis. In this paper, we provide bounds on the computational complexity of determining the interleaving distance in several settings. We show that the interleaving distance is NP-hard to compute for persistence modules valued in the category of vector spaces. In the specific setting of multidimensional persistent homology we show that the problem is at least as hard as a matrix invertibility problem. Furthermore, this allows us to conclude that the interleaving distance of interval decomposable modules depends on the characteristic of the field. Persistence modules valued in the category of sets are also studied. As a corollary, we obtain that the isomorphism problem for Reeb graphs is graph isomorphism complete.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases Persistent Homology, Interleavings, NP-hard

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.13

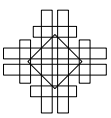
Related Version A full version of this paper is available at `https://arxiv.org/abs/1712.04281`

Acknowledgements This work was partially carried out while the authors were visitors to the Hausdorff Center for Mathematics, Bonn, during the special Hausdorff program on applied and computational topology. We thank anonymous referees for insightful comments.

1 Introduction

For a category $\mathcal{C}$ and a poset $\mathbf{P}$ we define a $\mathbf{P}$-*indexed (persistence) module valued in* $\mathcal{C}$ to a be a functor $M : \mathbf{P} \to \mathcal{C}$. We will denote the associated functor category by $\mathcal{C}^{\mathbf{P}}$. If $M, N \in \mathcal{C}^{\mathbf{P}}$ then M and N are of the same *type*. Such functors appear naturally in applications, and most commonly when $\mathbf{P} = \mathbf{R}^n$, n-tuples of real numbers under the normal product order, and $\mathcal{C} = \mathbf{Vec}_{\mathbb{K}}$, the category of vector spaces over the field $\mathbb{K}$, or $\mathcal{C} = \mathbf{Set}$, the category of sets. The field $\mathbb{K}$ is assumed to be finite. We suppress notation and simply write $\mathbf{Vec}$ when $\mathbb{K}$ is an arbitrary finite field. The notation $p \in \mathbf{P}$ denotes that p is an object of $\mathbf{P}$.

[1] DFG Collaborative Research Center SFB/TR 109 "Discretization in Geometry and Dynamics"



▶ Remark. Throughout the paper we make use of basic concepts from category theory. The reader unfamiliar to such ideas will find the necessary background material in the first few pages of [13].

Assume that $h : X \to \mathbb{R}$ is a continuous function of "Morse type", a generalization of a Morse function on a compact manifold. Roughly, a real-valued function is of Morse type if the homotopy type of the fibers changes at finite set of values; see [16] for a precise definition. We shall now briefly review four different scenarios in which functors of the aforementioned form can be associated to h.

Let $\mathrm{H}_p : \mathbf{Top} \to \mathbf{Vec}_{\mathbb{K}}$ denote the p-th singular homology functor with coefficients in $\mathbb{K}$, and let $\pi_0 : \mathbf{Top} \to \mathbf{Set}$ denote the functor giving the set of path-components. We also associate the two following functors to h whose actions on morphisms are given by inclusions:

$$\mathcal{S}^{\uparrow}(h) : \mathbf{R} \to \mathbf{Top} \qquad \mathcal{S}(h) : \mathbf{R}^2 \to \mathbf{Top}$$
$$\mathcal{S}^{\uparrow}(h)(t) = \{x \in X \mid h(x) \leq t\} \qquad \mathcal{S}(h)(-s,t) = \{x \in X \mid s \leq h(x) \leq t\}$$

- *Persistent Homology* studies the evolution of the homology of the sublevel sets of h and is perhaps the most prominent tool in topological data analysis [16]. Specifically, the *p-th sublevel set persistence module associated to h* is the functor $\mathrm{H}_p\mathcal{S}^{\uparrow}(h) : \mathbf{R} \to \mathbf{Vec}$. Importantly, such a module is completely determined by a collection of intervals $\mathcal{B}(\mathrm{H}_p\mathcal{S}^{\uparrow}(h))$ called the *barcode* of $\mathrm{H}_p\mathcal{S}^{\uparrow}(h)$. This collection of intervals is then in turn used to extract topological information from the data at hand. In Fig. 1 we show the associated barcode for $p = 0$ and $p = 1$ for a function of Morse type.
- Upon replacing H_p by π_0 in the above construction we get a *merge tree*. That is, the *merge tree associated to h* is the functor $\tau^h : \pi_0\mathcal{S}^{\uparrow}(h) : \mathbf{R} \to \mathbf{Set}$. A merge tree captures the evolution of the path components of the sublevel sets of h and can be, as the name indicates, be visualized as (a disjoint union of) rooted trees. See Fig. 1 for an example.
- The two aforementioned examples used sublevel sets. A richer invariant is obtained by considering interlevel sets: define the *p-th interlevel set persistence of h* to be the functor $\mathrm{H}_p\mathcal{S}(h) : \mathbf{R}^2 \to \mathbf{Vec}$. Analogously to above, such a module is completely determined by a collection $\mathcal{B}(\mathrm{H}_p\mathcal{S}(h))$ of simple regions in $\mathbb{R}^2$. However, it is often the collection of intervals $\mathcal{L}_p(h)$ obtained by the intersection of these regions with the anti-diagonal $y = -x$ which are used in data analysis. We refer the reader to [9] for an in-depth treatment. In Fig. 1 we show an example of the 0-th interlevel set barcode. Observe how the endpoints of the intervals correspond to different types of features of the Reeb graph.
- Just as interlevel set persistence is a richer invariant than sublevel set persistence, the *Reeb graph* is richer in structure than the merge tree. Specifically, we define the functor $\mathrm{Reeb}^h := \pi_0\mathcal{S}(h) : \mathbf{R}^2 \to \mathbf{Set}$. Just as for Merge trees, Reeb^h admits a visualization of a graph; see Fig. 1. In particular, this appealing representation has made Reeb graphs a popular objects of study in computational geometry and topology, and they have found many applications in data visualization and exploratory data analysis.

These are all examples of topological invariants arising from a single real-valued function. There are many settings for which it is more fruitful to combine a collection of real-valued functions into a single function $g : X \to \mathbb{R}^n$ [14]. By combining them into a single function we not only learn how the data looks from the point of view of each function (i.e. a type of measurement) but how the different functions (measurements) interact. One obvious way to assign a (algebraic) topological invariant to g is to filter it by sublevel sets. That is, define $\mathcal{S}^{\uparrow}(g) : \mathbf{R}^n \to \mathbf{Top}$ by $\mathcal{S}^{\uparrow}(g)(t) = \{x \in X \mid g(x) \leq t\}$. The associated functor $\mathrm{H}_p\mathcal{S}^{\uparrow}(g) : \mathbf{R}^n \to \mathbf{Vec}$ is an example of an *n-dimensional persistence module*. We saw above

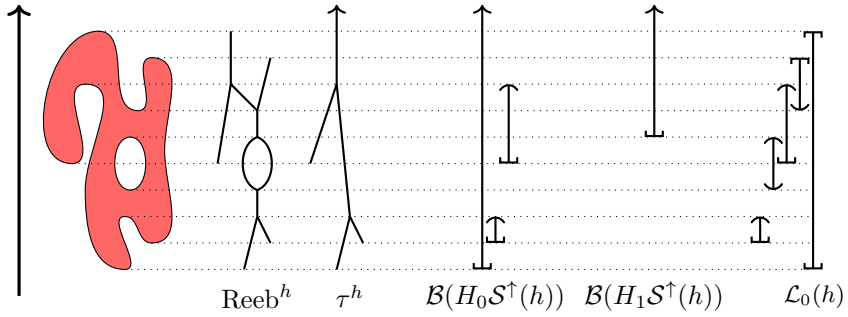


Figure 1 The height function of the solid shape is of Morse type. The associated Reeb graph, merge tree, sublevel set barcodes, and interlevel set barcode are shown to the right.

that for $n = 1$ this functor is completely described by a collection of intervals. This is far from true for $n \geq 2$: there exists no way to describe such functors by interval-like regions in higher-dimensional Euclidean space. Even the task of parameterizing such (indecomposable) modules is known to be a *hopeless* problem (so-called *wild representation type*) [3].

1.1 The interleaving distance

Different types of distances have been proposed on various types of persistence modules with values in **Vec** [26, 24, 17, 12, 5]. Of all these, the *interleaving distance* is arguably the most prominent for the following reasons: the theory of interleavings lies at the core of the theoretical foundations of 1-dimensional persistence, notably through the Isometry Theorem (Theorem 7). Furthermore, it was shown by Lesnick that when $\mathbb{K}$ is a prime field, the interleaving distance is the *most discriminative* of all *stable* metrics on such modules. We refer to [24] for the precise statement. As we shall see, it is also an immediate consequence of Theorem 7 that the interleaving distance for 1-dimensional persistence modules can be computed in polynomial time.

Lesnick's result generalizes to n-dimensional persistence modules, but the computational complexity of computing the interleaving distance of such modules remains unknown. An efficient algorithm to compute the interleaving distance could carry a profound impact on topological data analysis: the standard pipeline for 1-dimensional persistent homology is to first compute the barcode and then perform analysis on the collection of intervals. However, for multi-dimensional persistence there is no way of defining the barcode. With an efficient algorithm for computing the interleaving distance at hand it would still not be clear how to analyze the persistence modules individually, but we would have a theoretical optimal way of comparing them. This in turn could be used in clustering, kernel methods, and other kinds of data analysis widely applied in the 1-dimensional setting.

Complexity

The purpose of this paper is to determine the computational complexity of computing the interleaving distance. To make this precise, we need to associate a notion of size to the persistence modules.

▶ **Definition 1.** Let $\mathbf{P}$ denote a poset category and $M : \mathbf{P} \to \mathcal{C}$.

- For $\mathcal{C} = \mathbf{Vec}$, define the *total dimension of* M to be $\dim M = \sum_{p \in \mathbf{P}} \dim M_p$.
- For $M : \mathbf{Z} \to \mathbf{Set}$, define the *total cardinality of* M to be $|M| = \sum_{p \in \mathbf{P}} |M_p|$.

The input size will be the total dimension or the total cardinality and for the the remaining of the paper **we shall always assume** that those quantities are finite. The following shows that there exists an algorithm, polynomial in the input size, which determines whether or not two $\mathbf{P}$-indexed modules valued in $\mathbf{Vec}$ are isomorphic.

▶ **Theorem 2** ([10]). *Let* $\mathbf{P}$ *be a finite poset and* $M, M' : \mathbf{P} \to \mathbf{Vec}$. *There exists a deterministic algorithm which decides if* $M \cong M'$ *in* $\mathcal{O}\left((\dim M + \dim M')^6\right)$.

This result will be important to us in what ensues because the strongest of interleavings, the 0-interleaving, is by definition a pair of inverse isomorphisms. Furthermore, by choosing an appropriate basis for each vector space, an isomorphism between M and M' is nothing more than a collection of matrices with entries in a finite field. Likewise a δ-interleaving will be nothing more than a collection of matrices over a finite field satisfying certain constraints. When $\mathcal{C} = \mathbf{Set}$ the morphisms are specified by collections of functions between finite sets. Hence, the decision problems considered in this paper **are trivially in NP**.

Furthermore, it is an immediate property of the Morse type of h, that the modules considered above are *discrete*. Intuitively, we say that an $\mathbf{R}^n$-indexed persistence module M is discrete if there exists a $\mathbf{Z}^n$-indexed persistence module containing all the information of M; see [7]. In practice, persistence modules arising from data will be discrete. Hence, when it comes to algorithmic questions we shall restrict ourselves to the setting in which $\mathbf{P} = \mathbf{Z}^n$ or a slight generalization thereof. Importantly, the modules considered in this paper can be δ-interleaved only for $\delta \in \{0, 1, 2, \ldots\}$.

Contributions

The contributions of this paper are summarized in Table 1. Concretely, a cell in Table 1 gives a complexity bound on the decision problem of deciding if two modules of the given type are δ-interleaved. It is an easy consequence of the definition of the interleaving distance that this is at least as hard as determining the distance itself. The cells with a shaded background indicate that novel contributions to that complexity bound is provided in this paper. Recall that we have defined the input size to be $n = \dim M + \dim M'$ when the modules are valued in $\mathbf{Vec}$, and $n = |M| + |M'|$ when the modules are valued in $\mathbf{Set}$. Observe that any non-trivial functor $M : \mathbf{Z}^m \to \mathbf{Set}$ must have $|M| = \infty$. Hence, when we talk about interleavings of such functors, we shall assume that they are completely determined by a restriction to a finite sub-grid. The input size is then the total cardinalities of the restrictions. We will now give a brief summary of the cells of Table 1.

- $\mathbf{Z} \to \mathbf{Vec}$. $[\delta \geq 0]$ This bound is achieved by first determining the barcodes of the persistence modules and then using Theorem 7 to obtain the interleaving distance. The complexity of this is $\mathcal{O}(\text{FindBarcode} + \text{Match}) = \mathcal{O}(n^\omega + n^{1.5} \log n) = \mathcal{O}(n^\omega)$ where ω is the matrix multiplication exponent[22]. The details can be found in [7]. In [25], the complexity is shown to be $\mathcal{O}(n^\omega + n^2 \log^2 n)$ for essentially the same problem, but with a slightly different input size n.
- $\mathbf{Z} \to \mathbf{Set}$. $[\delta = 0]$ Essentially isomorphism of rooted trees; see [7]. $[\delta \geq 1]$ This follows from arguments in [1].
- $\mathbf{Z}^2 \to \mathbf{Vec}$. $[\delta = 0]$ This is Theorem 2 for $\mathbf{P} = \mathbf{Z}^2$. $[\delta \geq 1]$ A *constrained invertibility* (CI) problem is a triple (P, Q, n) where P and Q are subsets of $\{1, 2, \ldots, n\}^2$. We say that a CI-problem (P, Q, n) is *solvable* if there exists an invertible $n \times n$ matrix M such that $M_{i,j} = 0$ for all $(i, j) \in P$ and $M^{-1}_{i',j'} = 0$ for all $(i', j') \in Q$. We call (M, M^{-1}) a *solution* of (P, Q, n). In Section 4 we show that a CI-problem is solvable if and only if an associated pair of $\mathbf{Z}^2$-indexed modules is 1-interleaved. Thus, the interleaving problem is *constrained invertibility-hard* (CI-hard).

Table 1 The complexity of checking for δ-interleavings between modules M and M'. If the target category is **Vec** then $n = \dim M + \dim M'$, and if the target category is **Set** then $n = |M| + |M'|$. Here ω is the matrix multiplication exponent.

type/δ	$\mathbf{Z} \to \mathbf{Vec}$	$\mathbf{Z} \to \mathbf{Set}$	$\mathbf{Z}^2 \to \mathbf{Vec}$	$\mathbf{Z}^2 \to \mathbf{Set}$	$\mathbf{Z}^{L,C} \to \mathbf{Vec}_{\mathbb{Z}/2\mathbb{Z}}$
$\delta = 0$	$\mathcal{O}(n^\omega)$	$\mathcal{O}(n)$	$\mathcal{O}(n^6)$	GI-complete	$\mathcal{O}(n^6)$
$\delta \geq 1$	$\mathcal{O}(n^\omega)$	NP-complete	CI-hard	NP-complete	NP-complete

- $\mathbf{Z}^2 \to \mathbf{Set}$. $[\delta = 0]$ Reeb graphs are a particular type of functors $\mathbf{Z}^2 \to \mathbf{Set}$ and deciding if two Reeb graphs are isomorphic is graph isomorphism-hard (GI-hard) [20]. In [7] we strengthen this result by showing that the isomorphism problem for $\mathbf{Z}^2 \to \mathbf{Set}$ is in fact GI-*complete*. This also implies that Reeb graph isomorphism is GI-complete. $[\delta \geq 1]$ This follows from $\mathbf{Z} \to \mathbf{Set}$.
- $\mathbf{Z}^{L,C} \to \mathbf{Vec}_{\mathbb{Z}/2\mathbb{Z}}$. For two sets L and C, define $\mathbf{Z}^{L \to C}$ to be the poset generated by the following disjoint union of posets $\mathbf{Z}^{L \to C} := \bigsqcup_{l \in L, c \in C} \mathbf{Z}$ with the added relation $(l, t) < (c, t)$ for every $l \in L$, $c \in C$ and $t \geq 3$. This poset is a mild generalization of a disjoint union of $\mathbf{Z}$'s. $[\delta = 0]$ Immediate from Theorem 2. $[\delta \geq 1]$ Follows from a reduction from 3-SAT; see Section 3. This shows that computing the generalized interleaving distance of [12] for **Vec**-valued persistence modules is NP-complete in general.

2 Preliminaries

For $\mathbf{P}$ a poset and $\mathcal{C}$ an arbitrary category, $M : \mathbf{P} \to \mathcal{C}$ a functor, and $a, b \in \mathbf{P}$, let $M_a = M(a)$, and let $\varphi_M(a, b) : M_a \to M_b$ denote the morphism $M(a \leq b)$.

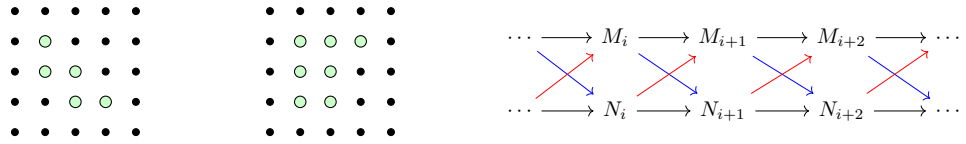
2.1 Interleavings

In this section we review the theory of interleavings for $\mathbf{Z}^n$-indexed modules. For a treatment of the $\mathbf{R}^n$-indexed setting see [24]. For a discussion on interleavings over arbitrary posets see [12].

For $u \in \mathbf{Z}^n$, define the *u-shift functor* $(-)(u) : \mathcal{C}^{\mathbf{Z}^n} \to \mathcal{C}^{\mathbf{Z}^n}$ on objects by $M(u)_a = M_{u+a}$, together with the obvious internal morphisms, and on morphisms $f : M \to N$ by $f(u)_a = f(u + a) : M(u)_a \to N(u)_a$. For $u \in \{0, 1, \ldots\}^n$, let $\varphi_M^u : M \to M(u)$ be the morphism whose restriction to each M_a is the linear map $\varphi_M(a, a + u)$. For $\delta \in \{0, 1, 2 \ldots\}$ we will abuse notation slightly by letting $(-)(\delta)$ denote the $\delta(1, \ldots, 1)$-shift functor, and letting φ_M^δ denote $\varphi_M^{\delta(1,\ldots,1)}$.

▶ **Definition 3.** Given $\delta \in \{0, 1, \ldots\}$, a δ-interleaving between $M, N : \mathbf{Z}^n \to \mathcal{C}$ is a pair of morphisms $f : M \to N(\delta)$ and $g : N \to M(\delta)$ such that $g(\delta) \circ f = \varphi_M^{2\delta}$ and $f(\delta) \circ g = \varphi_N^{2\delta}$.

We call f and g *δ-interleaving morphisms*. If there exists a δ-interleaving between M and N, we say M and N are δ-interleaved. The *interleaving distance* $d_I : \mathrm{Ob}(\mathcal{C}^{\mathbf{Z}^n}) \times \mathrm{Ob}(\mathcal{C}^{\mathbf{Z}^n}) \to [0, \infty]$ is given by $d_I(M, N) = \min\{\delta \in \{0, 1, \ldots\} \mid M \text{ and } N \text{ are } \delta\text{-interleaved}\}$. Here we set $d_I(M, N) = \infty$ if there does not exist a δ-interleaving for any δ.



(a) An interval. **(b)** Not an interval. **(c)** A 1-interleaving.

Figure 2 (a) is an interval in $\mathbf{Z}^2$ whereas (b) is not. (c) The persistence modules M and N are 1-interleaved if and only if there exist diagonal morphisms such that the diagram in (c) commutes.

2.2 Interval modules and the Isometry Theorem

Let $\mathcal{C} = \mathbf{Vec}$. An *interval* of a poset $\mathbf{P}$ is a subset $\mathcal{J} \subset \mathbf{P}$ such that

1. $\mathcal{J}$ is non-empty.
2. If $a, c \in \mathcal{J}$ and $a \leq b \leq c$, then $b \in \mathcal{J}$.
3. [connectivity] For any $a, c \in \mathcal{J}$, there is a sequence $a = b_0, b_1, \ldots, b_l = c$ of elements of $\mathcal{J}$ with b_i and b_{i+1} comparable for $0 \leq i \leq l-1$.

We refer to a collection of intervals in $\mathbf{P}$ as a *barcode (over* $\mathbf{P}$*)*.

▶ **Definition 4.** For $\mathcal{J}$ an interval in $\mathbf{P}$, the interval module $I^{\mathcal{J}}$ is the $\mathbf{P}$-indexed module such that

$$I^{\mathcal{J}}_a = \begin{cases} \mathbb{K} & \text{if } a \in \mathcal{J}, \\ 0 & \text{otherwise.} \end{cases} \qquad \varphi_{I^{\mathcal{J}}}(a,b) = \begin{cases} \mathrm{id}_{\mathbb{K}} & \text{if } a \leq b \in I, \\ 0 & \text{otherwise.} \end{cases}$$

We say a persistence module M is *decomposable* if it can be written as $M \cong V \oplus W$ for non-trivial persistence modules V and W; otherwise, we say that M is *indecomposable*.

A $\mathbf{P}$-indexed module M is *interval decomposable* if there exists a collection $\mathcal{B}(M)$ of intervals in $\mathbf{P}$ such that $M \cong \bigoplus_{\mathcal{J} \in \mathcal{B}(M)} I^{\mathcal{J}}$. We call $\mathcal{B}(M)$ the *barcode* of M. This is well-defined by the Azumaya–Krull–Remak–Schmidt theorem [2].

▶ **Theorem 5** (Structure of 1-D Modules [19, 27])**.** *Suppose* $M : \mathbf{P} \to \mathbf{Vec}$ *for* $\mathbf{P} \in \{\mathbf{R}, \mathbf{Z}\}$ *and* $\dim M_p < \infty$ *for all* $p \in \mathbf{P}$*. Then* M *is interval decomposable.*

▶ Remark. Such a decomposition theorem exists only for *very* special choices of $\mathbf{P}$. Two other scenarios appearing in applications are *zigzags* [8, 15] and exact bimodules [18]. The latter is a specific type of $\mathbf{R}^2$-indexed persistence modules.

▶ **Corollary 6.** *Let* $\mathbf{P} = \bigsqcup_{i \in \Lambda} \mathbf{Z}$ *be the poset given as a disjoint union of* $\mathbf{Z}$*'s (i.e. elements in different components are incomparable). If* $M : \mathbf{P} \to \mathbf{Vec}$ *satisfies* $\dim M_p < \infty$ *for all* $p \in \mathbf{P}$*, then* M *is interval decomposable.*

Proof. Apply Theorem 5 to each of the components of $\mathbf{P}$ independently. This gives $\mathcal{B}(M) = \bigsqcup_{i \in \Lambda} \mathcal{B}(M|_{(i,\mathbf{Z})})$. ◀

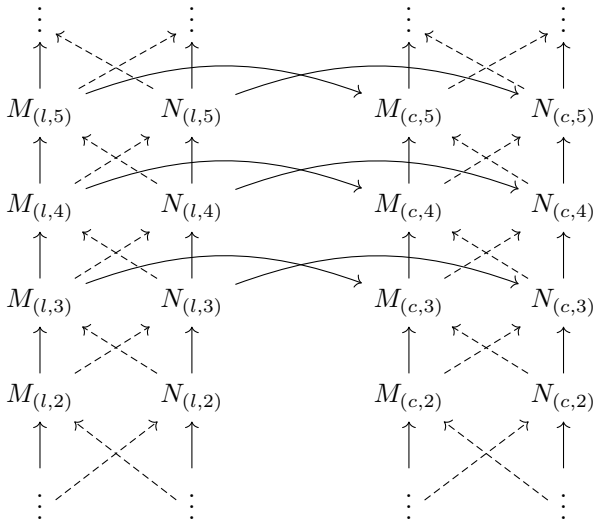
At the very core of topological data analysis are the *isometry theorems*. They say that for certain choices of interval decomposable modules, the interleaving distance coincides with a completely combinatorial distance on their associated barcodes. This combinatorial distance d_B is called the *bottleneck distance* and is defined in [7]. Importantly, for any two barcodes, if the interleaving distance between each pair of interval modules in the barcodes is known, the associated bottleneck distance can be computed by solving a bipartite matching problem. This, in turn, implies that the interleaving distance can be efficiently computed whenever an isometry theorem holds. See [7] for an example.

▶ **Theorem 7** (Isometry Theorem [24, 17, 4, 6]). *Suppose* $M, N : \mathbf{Z} \to \mathbf{Vec}$ *satisfy* $\dim M_i < \infty$ *and* $\dim N_i < \infty$ *for all* $i \in \mathbf{Z}$. *Then* $d_I(M, N) = d_B(\mathcal{B}(M), \mathcal{B}(N))$.

▶ Remark. Continuing on the remark to Theorem 5. An isometry theorem also holds for zigzags and exact bimodules [9, 6]. Although there might be other classes of interval decomposable modules for which an isometry theorem holds, the result is not true in general. See [7] for an example of interval decomposable modules in $\mathbf{Z}^2$ for which $2d_I(M, N) = d_B(\mathcal{B}(M), \mathcal{B}(N))$, and see [9] for a general conjecture. This shows that a matching of the barcodes will not determine the interleaving distance even in the case of very well-behaved modules.

3 NP-completeness

In this section we shall prove that it is NP-hard to decide if two modules $M, N \in \mathbf{Vec}^{\mathbf{Z}^{L\to C}}$ are 1-interleaved. Recall that for two sets L and C, we define $\mathbf{Z}^{L\to C}$ to be the disjoint union $\bigsqcup_{l\in L, c\in C} \mathbf{Z}$ with the added relations $(l, t) < (c, t)$ for all $l \in L, c \in C$, and $t \geq 3$. Define the *u-shift functor* $(-)(u) : \mathcal{C}^{\mathbf{Z}^{L\to C}} \to \mathcal{C}^{\mathbf{Z}^{L\to C}}$ on objects by $M(u)_{(p,t)} = M_{(p,t+u)}$, together with the obvious internal morphisms, and on morphisms $f : M \to N$ by $f(u)_{(p,t)} = f_{(p,t+u)} : M(u)_{(p,t)} \to N(u)_{(p,t)}$. That is, the shift functor simply acts on each of the components independently. With the shift-functor defined, we define a δ-interleaving of $\mathbf{Z}^{L\to C}$-indexed modules precisely as in Section 2.1. Thus, we see that a δ-interleaving is simply a collection of δ-interleavings over each disjoint component of $\mathbf{Z}$ which satisfy the added relations. Indeed, a 1-interleaving is equivalent to the existence of dashed morphisms in the following diagram for all $l \in L$ and $c \in C$:



We saw in Theorem 6 that $M : \mathbf{Z}^{L\to\emptyset} \to \mathbf{Vec}$ is interval decomposable. By applying Theorem 7 to each disjoint component independently, the following is easy to show. Here the bottleneck distance is generalized in the obvious way, i.e. matching each component independently.

▶ **Corollary 8** (Isometry Theorem for Disjoint Unions). *Let* L *be any set, and* $M, N : \mathbf{Z}^{L\to\emptyset} \to \mathbf{Vec}$ *such that* $\dim M_p < \infty$ *and* $\dim N_p < \infty$ *for all* $p \in \mathbf{Z}^{L\to\emptyset}$. *Then* $d_I(M, N) = d_B(\mathcal{B}(M), \mathcal{B}(N))$.

In particular, the interleaving distance between M and N can be effectively computed through a bipartite matching. As we shall see, this is not true for $C \neq \emptyset$. The remainder of this section is devoted to proving the following theorem:

▶ **Theorem 9.** *Unless P=NP, there exists no algorithm, polynomial in $n = \dim M + \dim N$, which decides if $M, N : \mathbf{Z}^{L \to C} \to \mathbf{Vec}_{\mathbb{Z}/2\mathbb{Z}}$ are 1-interleaved.*

3.1 The proof

We shall prove Theorem 9 by a reduction from 3-SAT. Let ψ be a boolean formula in 3-CNF defined on literals $L = \{x_1, x_2, \ldots, x_{n_l}\}$ and clauses $C = \{c_1, c_2, \ldots, c_{n_c}\}$. We shall assume that the literals of each clause are distinct and *ordered*. That is, the clause c_i is specified by the three distinct literals $\{x_{i_1}, x_{i_2}, x_{i_3}\}$ wherein $i_1 < i_2 < i_3$. Determining if ψ is satisfiable is well-known to be NP-complete. For the entirety of the proof $\mathbb{K} = \mathbb{Z}/2\mathbb{Z}$.

Step 1: Defining the representations. Associate to ψ two functors $M, N : \mathbf{Z}^{L \to C} \to \mathbf{Vec}_{\mathbb{Z}/2\mathbb{Z}}$ in the following way: For all literals $x_j \in L$ define

$$M_{(x_j,1)} \longrightarrow M_{(x_j,2)} \longrightarrow M_{(x_j,3)} \longrightarrow M_{(x_j,4)} = \mathbb{K} \xrightarrow{1} \mathbb{K} \xrightarrow{1} \mathbb{K} \xrightarrow{1} \mathbb{K}$$

$$N_{(x_j,1)} \longrightarrow N_{(x_j,2)} \longrightarrow N_{(x_j,3)} \longrightarrow N_{(x_j,4)} = \mathbb{K} \xrightarrow{1} \mathbb{K} \xrightarrow{(1;1)} \mathbb{K}^2 \longrightarrow 0$$

and for every clause c_i in ψ define

$$M_{(c_i,1)} \longrightarrow M_{(c_i,2)} \longrightarrow M_{(c_i,3)} \longrightarrow M_{(c_i,4)} = 0 \longrightarrow \mathbb{K} \xrightarrow{1} \mathbb{K} \xrightarrow{1} \mathbb{K}$$

$$N_{(c_i,1)} \longrightarrow N_{(c_i,2)} \longrightarrow N_{(c_i,3)} \longrightarrow N_{(c_i,4)} = 0 \longrightarrow \mathbb{K}^3 \xrightarrow{1} \mathbb{K}^3 \longrightarrow 0$$

For any other $p \in \mathbf{Z}^{L \to C}$, $M_p = N_p = 0$. Next we specify the remaining non-trivial morphisms: let $c_i = z_{i_1} \vee z_{i_2} \vee z_{i_3}$ be a clause in ψ, where $z_{j_l} = x_{j_l}$ or $z_{j_l} = \neg x_{j_l}$, and for which $i_1 < i_2 < i_3$. For $s = 1, 2, 3$ define $H_s : \mathbb{K}^2 \to \mathbb{K}^3$ by $e_1 \to u \cdot e_s$ and $e_2 \to (1-u) \cdot e_s$, where $u = 1$ if $z_{i_s} = x_{i_s}$ and $u = 0$ if $z_{i_s} = \neg x_{i_s}$. Here e_d is the d-th standard basis vector of $\mathbb{K}^3$. Given this we define the following for $s = 1, 2, 3$:

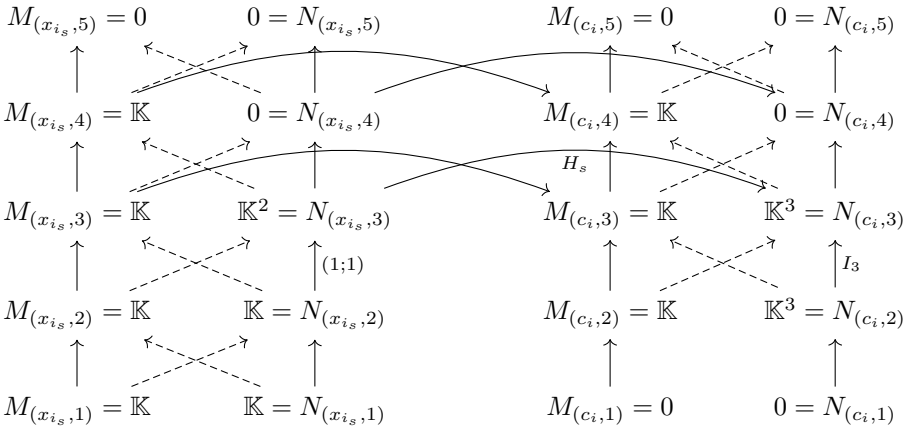
$$\begin{array}{ccc} M_{(x_{i_s},4)} \longrightarrow M_{(c_i,4)} & = & \mathbb{K} \xrightarrow{1} \mathbb{K} \\ \uparrow \qquad\quad \uparrow & & {\scriptstyle 1}\uparrow \qquad \uparrow{\scriptstyle 1} \\ M_{(x_{i_s},3)} \longrightarrow M_{(c_i,3)} & = & \mathbb{K} \xrightarrow{1} \mathbb{K} \end{array}$$

and

$$N_{(x_{i_s},3)} \longrightarrow N_{(c_i,3)} = \mathbb{K}^2 \xrightarrow{H_s} \mathbb{K}^3.$$

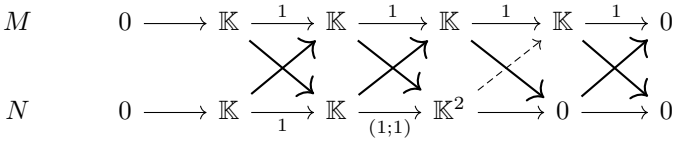
Clearly $\dim M + \dim N = \mathcal{O}(n_c + n_l)$. Thus, the total dimension is polynomial in the input size of 3-SAT.

Step 2: Showing the reduction. Observe that M and N are 1-interleaved if and only if there exist dashed morphisms such that the below diagram is commutative for every literal x_{i_s} and for every clause c_i containing x_{i_s}:

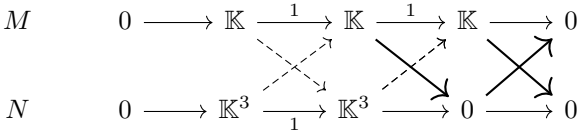


(1)

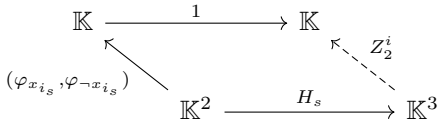
We shall see there are few degrees of freedom in the choice of interleaving morphisms. Indeed, consider the left part of the above diagram:



We leave it to the reader to verify that if M and N are 1-interleaved, then all the solid diagonal morphisms in the above diagram are completely determined by commutativity. For the dashed morphism $(\varphi_{x_{i_s}}, \varphi_{\neg x_{i_s}}) : \mathbb{K}^2 \to \mathbb{K}$ there are two choices: by commutativity it must satisfy $(\varphi_{x_{i_s}}, \varphi_{\neg x_{i_s}}) \cdot (1;1) = 1$ and thus $\varphi_{x_{i_s}} + \varphi_{\neg x_{i_s}} = 1$. As $\mathbb{K} = \mathbb{Z}/2\mathbb{Z}$, this implies that precisely one of $\varphi_{x_{i_s}}$ and $\varphi_{\neg x_{i_s}}$ is multiplication by 1. This corresponds to a choice of truth value for x_{i_s}: $\varphi_{x_{i_s}} = 1 \iff x_{i_s} = \text{True}$ and $\varphi_{\neg x_{i_s}} = 1 \iff x_{i_s} = \text{False}$. Next, consider the right part of 1:



There are three non-trivial morphisms, out of which two are equal by commutativity. Let $Z_1^i : \mathbb{K} \to \mathbb{K}^3$ and $Z_2^i : \mathbb{K}^3 \to \mathbb{K}$ denote the two unspecified morphisms. Returning to (1), we see that Z_2^i must satisfy the following for $s \in \{1, 2, 3\}$:



Thus, Z_2^i restricted to its s-th component equals either $\varphi_{x_{i_s}}$ or $\varphi_{\neg x_{i_s}}$, depending on whether x_{i_s} or its negation $\neg x_{i_s}$ appears in the clause c_i. This implies that Z_2^i is given by

$$Z_2^i = \begin{bmatrix} \varphi_{z_{i_1}} & \varphi_{z_{i_2}} & \varphi_{z_{i_3}} \end{bmatrix}$$

Hence, if M and N are to be 1-interleaved, then there are no degrees of freedom in choosing Z_2^i after the $\varphi_{x_{i_s}}$ are specified. However, Z_1^i only needs to satisfy $Z_2^i \circ Z_1^i = 1$. As this is the sole restriction imposed on Z_1^i, we see that this can be satisfied if and only if $Z_2^i \neq 0$, which is true if and only if $z_{i_s} = \text{True}$ for at least one $s \in \{1, 2, 3\}$.

▶ **Theorem 10.** *Let ψ be a boolean formula as above. Then ψ is satisfiable if and only if the associated persistence modules $M, N : \mathbf{Z}^{L \to C} \to \mathbf{vec}$ are 1-interleaved.*

Proof. Summarizing the above: we have that M and N are 1-interleaved if and only if we can choose morphisms $(\varphi_{x_{i_s}}, \varphi_{\neg x_{i_s}})$ such that $Z_2^i \neq 0$ for all clauses c_i. This means precisely that we can choose truth values for each x_{i_s} such that every clause $c_i = z_{i_1} \vee z_{i_2} \vee z_{i_3}$ evaluates to true. This shows that a 1-interleaving implies that ψ is satisfiable. Conversely, if ψ is satisfiable, then we see that the morphisms defined by $\varphi_{x_{i_s}} = 1 \iff x_{i_s} = \text{True}$ and $\varphi_{\neg x_{i_s}} = 1 \iff x_i = \text{False}$ satisfy $Z_2^i \neq 0$ for every clause c_i. Thus, M and N are 1-interleaved. ◀

▶ Remark. Let $i : \mathbf{P} \hookrightarrow \mathbf{Q}$ be an inclusion of posets and $M : \mathbf{P} \to \mathbf{Vec}$. There are multiple functorial ways of extending M to a representation $E(M) : \mathbf{Q} \to \mathbf{Vec}$, e.g. by means of left or right Kan extensions. This is a key ingredient in one of the more recent proofs of Theorem 7; see [11] for details. However, if we impose the condition that $E(M) \circ i \cong M$ then such an extension need not exist. Indeed, Theorem 10 implies that the associated decision problem is NP-complete.

4 Interleavings of multidimensional persistence modules

Recall that a constrained invertibility (CI) problem is a triple (P, Q, n) where P and Q are subsets of $\{1, 2, \ldots, n\}^2$, and that a CI-problem is solvable if there exists an invertible $n \times n$ matrix M such that $M_{(i,j)} = 0$ for all $(i, j) \in P$ and $M^{-1}_{(i',j')} = 0$ for all $(i', j') \in Q$. We shall show that a CI-problem is solvable if and only if a pair of associated persistence modules $\mathbf{Z}^2 \to \mathbf{Vec}$ is 1-interleaved. Hence, if deciding solvability is NP-hard, then so is computing the interleaving distance for multidimensional persistence modules.

▶ **Example 11.** Let $P = \{(2, 2), (3, 3)\}, Q = \{(2, 3), (3, 2)\} \subset \{1, 2, 3\}^2$. Then $(P, Q, 3)$ is solvable by

$$M = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}, \quad M^{-1} = \begin{bmatrix} -1 & 1 & 1 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \end{bmatrix}.$$

▶ **Example 12.** Let $P = \{(1, 1), (1, 3)\}, Q = \{(2, 1)\} \subset \{1, 2, 3\}^2$. Then $(P, Q, 3)$ is not solvable, as $(MN)_{(1,1)} = 0$ for all 3×3-matrices M, N with $M_{(1,1)} = M_{(1,3)} = N_{(2,1)} = 0$. Note that it matters that we view P and Q as subsets of $\{1, 2, 3\}^2$ and not of $\{1, \ldots, n\}^2$ for some $n > 3$, in which case (P, Q) would be solvable.

▶ **Example 13.** Observe that a CI-problem $(P, \emptyset, n)$ reduces to a bipartite matching problem. Build a graph G on $2n$ vertices $\{v_1, \ldots, v_n, u_1, \ldots, u_n\}$ with an edge from v_i to u_j if $(i, j) \notin P$. Then the CI-problem is solvable if and only if there exists a perfect matching of G.

A CI-problem can be seen as a problem of choosing weights for the edges in a directed simple graph: Given (P, Q, n), let G be the bipartite directed simple graph with vertices $\{u_1, \ldots, u_n, v_1, \ldots, v_n\}$, an edge from u_i to v_j if $(i, j) \notin P$, and an edge from v_j to u_i if $(j, i) \notin Q$. Solving (P, Q, n) is then equivalent to weighting the edges in G with elements from $\mathbb{K}$ so that

$$\sum_{j=1}^{n} w(u_i, v_j) w(v_j, u_i) = 1$$

for all i, and

$$\sum_{j=1}^{n} w(u_i, v_j)w(v_j, u_{i'}) = 0$$

for all $i \neq i'$, where $w(u, v)$ is the weight of the edge from u to v if there is one, and 0 if not. If the weights are elements of $\mathbb{Z}/2\mathbb{Z}$, this is equivalent to picking a subset of the edges such that there is an odd number of paths of length two from any vertex to itself and an even number of paths of length two from any vertex to any other vertex.

Fix a CI-problem (P, Q, n) and let $m = |P| + |Q|$. We will construct $\mathbf{Z}^2$-indexed modules M and N that are 1-interleaved if and only if (P, Q, n) is solvable, and that are zero outside a grid of size $(2m+3) \times (2m+3)$ in $\mathbf{Z}^2$. The dimension of each vector space $M_{(a,b)}$ or $N_{(a,b)}$ is bounded by n, so the total dimensions of M and N are polynomial in n.

For $p \in \mathbf{Z}^2$, let $\langle p \rangle = \{q \in \mathbf{Z}^2 \mid p \leq q \leq (2m+2, 2m+2)\}$. Let $\mathcal{W}$ be the interval $\bigcup_{k=0}^{m} \langle (2m-2k, 2k) \rangle$, and for $i \in \{1, 2, \ldots, m\}$, let $x_i = (2m-2i+1, 2i-1)$; see Fig. 3.

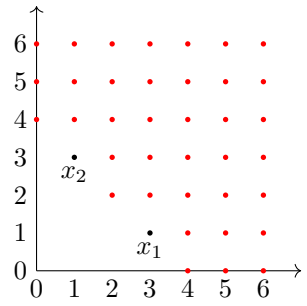


Figure 3 The interval $\mathcal{W}$ for $m = 2$ along with $x_1 = (3, 1)$ and $x_2 = (1, 3)$.

Write $P = \{(p_1, q_1), \ldots, (p_r, q_r)\}$ and $Q = \{(p_{r+1}, q_{r+1}), \ldots, (p_m, q_m)\}$. We define $M = \bigoplus_{i=1}^{n} I^{\mathcal{I}_i}$ and $N = \bigoplus_{i=1}^{n} I^{\mathcal{J}_i}$, where $\mathcal{I}_i$ and $\mathcal{J}_i$ are constructed as follows: let $\mathcal{I}_i^0 = \mathcal{J}_i^0 = \mathcal{W}$ for all i. For $k = 1, 2, \ldots, r$, let

$$\mathcal{I}_i^k = \begin{cases} \mathcal{I}_i^{k-1} \cup \langle x_k - (1,1) \rangle, & \text{if } i = p_k \\ \mathcal{I}_i^{k-1} \cup \langle x_k \rangle, & \text{if } i \neq p_k \end{cases}, \qquad \mathcal{J}_i^k = \begin{cases} \mathcal{J}_i^{k-1}, & \text{if } i = q_k \\ \mathcal{J}_i^{k-1} \cup \langle x_k \rangle, & \text{if } i \neq q_k \end{cases}$$

and for $k = r+1, \ldots, m$, let

$$\mathcal{I}_i^k = \begin{cases} \mathcal{I}_i^{k-1}, & \text{if } i = q_k \\ \mathcal{I}_i^{k-1} \cup \langle x_k \rangle, & \text{if } i \neq q_k \end{cases}, \qquad \mathcal{J}_i^k = \begin{cases} \mathcal{J}_i^{k-1} \cup \langle x_k - (1,1) \rangle, & \text{if } i = p_k \\ \mathcal{J}_i^{k-1} \cup \langle x_k \rangle, & \text{if } i \neq p_k \end{cases}$$

and let $\mathcal{I}_i = \mathcal{I}_i^m$ and $\mathcal{J}_i = \mathcal{J}_i^m$. This way, we ensure that there is no nonzero morphism from $I^{\mathcal{I}_i}$ to $I^{\mathcal{J}_j}(1)$ when $(i, j) \in P$, and no nonzero morphism from $I^{\mathcal{J}_j}$ to $I^{\mathcal{I}_i}(1)$ when $(j, i) \in Q$. In all other cases, there exist nonzero morphisms.

▶ **Lemma 14.** *Suppose $(i, j) \notin P$. Then there is an isomorphism $\mathrm{Hom}(I^{I_i}, I^{\mathcal{J}_j}(1)) \cong \mathbb{K}$. In particular, any morphism $f \in \mathrm{Hom}(I^{I_i}, I^{\mathcal{J}_j}(1))$ is completely determined by $f_{(2m+1,2m+1)}$: if f_p is nonzero, then $f_p = f_{(2m+1,2m+1)}$.*

The same holds if $(j, i) \notin Q$ instead of $(i, j) \notin P$, and $\mathcal{I}_i$ and $\mathcal{J}_j$ are interchanged. As f_p is a $\mathbb{K}$-endomorphism, this implies that any f can be identified with an element of $\mathbb{K}$.

Proof. Let $f : I^{\mathcal{I}_i} \to I^{\mathcal{J}_j}(1)$ be nonzero. If $p \notin \mathcal{I}_i$ or $p \not\leq (2m+1, 2m+1)$, $f_p = 0$. For $(2m+1, 2m+1) \geq p \in \mathcal{I}_i$, we have $p + (1,1) \in \mathcal{J}_i$ by construction and the fact that $(i, j) \notin P$, so $\varphi_{I^{\mathcal{J}_j}}(p + (1,1), (2m+2, 2m+2))$ is nonzero and hence the identity. We get

$$\begin{aligned} f_p &= \varphi_{I^{\mathcal{J}_j}(1)}(p, (2m+1, 2m+1)) \circ f_p \\ &= f_{(2m+1,2m+1)} \circ \varphi_{I^{\mathcal{I}_i}}(p, (2m+1, 2m+1)) = f_{(2m+1,2m+1)}. \end{aligned}$$

◀

Describing a morphism from $M = \bigoplus_{i=1}^{n} I^{\mathcal{I}_i}$ to $N(1) = \bigoplus_{j=1}^{n} I^{\mathcal{J}_j}(1)$ is the same as describing morphisms from $I^{\mathcal{I}_i}$ to $I^{\mathcal{J}_j}(1)$ for all i and j[2]. We have just proved that these can be identified with elements of $\mathbb{K}$, so we conclude that any $f : M \to N(1)$ is uniquely defined by an $n \times n$-matrix A_f where the entry (i, j) is the element in $\mathbb{K}$ corresponding to the morphism $I^{\mathcal{I}_i} \to I^{\mathcal{J}_j}(1)$ given by f. Note that we get the same result by writing $f_{(2m,2m)} = f_{(2m+1,2m+1)} : \mathbb{K}^n \to \mathbb{K}^n$ as a matrix, where each copy of $\mathbb{K}$ in the domain and codomain comes from one of the interval modules $I^{\mathcal{I}_i}$ or $I^{\mathcal{J}_j}(1)$, respectively.

If we also have a morphism $g : N \to M(1)$, we can define a matrix A_g symmetrically, and similarly A_g is $g_{(2m,2m)} = g_{(2m+1,2m+1)} : \mathbb{K}^n \to \mathbb{K}^n$ in matrix form.

▶ **Theorem 15.** *With f and g as above, (f, g) is a 1-interleaving if and only if A_f and A_g are inverse matrices.*

Proof. Suppose (f, g) is a 1-interleaving. The internal morphism $\varphi_M((2m, 2m), (2m+2, 2m+2))$ is the identity on $\mathbb{K}^n$, and is by definition of interleaving the same as

$$g(1)_{(2m,2m)} \circ f_{(2m,2m)} = g_{(2m+1,2m+1)} \circ f_{(2m,2m)} = A_g A_f.$$

Thus $A_g A_f$ is the identity matrix, and so A_f and A_g are inverses of each other, as both are $n \times n$-matrices.

Suppose A_f and A_g are inverse matrices. We must check that at every point $p \in \mathbb{Z}^2$, $\varphi_M(p, p + (2, 2)) = g(1)_p \circ f_p$. If $p \not\leq (2m, 2m)$ or $M(p) = 0$, both sides are zero. If $p \leq (2m, 2m)$ and $M(p) \neq 0$,

$$g(1)_p \circ f_p = \varphi_M(p + (2, 2), (2m + 2, 2m + 2)) \circ g(1)_p \circ f_p,$$

since $\varphi_M(p + (2, 2), (2m + 2, 2m + 2))$ must be the identity by construction of M and the fact that $M(p) \neq 0$. This is equal to

$$\begin{aligned}
&\varphi_M(p + (2, 2), (2m + 2, 2m + 2)) \circ g_{p+(1,1)} \circ f_p \\
&= g_{(2m+1,2m+1)} \circ \varphi_N(p + (1, 1), (2m + 1, 2m + 1)) \circ f_p \\
&= g_{(2m+1,2m+1)} \circ f_{(2m,2m)} \circ \varphi_M(p, (2m, 2m)) \\
&= A_g A_f \circ \varphi_M(p, (2m, 2m)) = \varphi_M(p, (2m, 2m)) = \varphi_M(p, p + (2, 2)).
\end{aligned}$$

◀

We have proved that defining morphisms $f : M \to N(1)$ and $g : N \to M(1)$ is the same as choosing $n \times n$-matrices A_f and A_g such that the entries corresponding to the elements of P and Q are zero, and that (f, g) is a 1-interleaving if and only if A_f and A_g are inverse matrices. Thus M and N are 1-interleaved if and only if the CI-problem (P, Q, n) is solvable.

We constructed M and N by setting all the interval modules comprising M and N equal to $I^{\mathcal{W}}$, then modifying them in m steps each, where the complexity of each step is clearly polynomial in n. Thus the complexity of constructing M and N is polynomial in n, and so are the total dimensions of M and N. Taking n^2 as the input complexity of solving a CI-problem (P, Q, n), we have proved a reduction implying the following theorem:

▶ **Theorem 16.** *Determining the interleaving distance for modules $\mathbf{Z}^2 \to \mathbf{Vec}$ is CI-hard.*

▶ Remark. We give an example in [7] of a CI-problem whose associated matrices M and N satisfy $d_I(M, N) = 1$ and $d_B(\mathcal{B}(M), \mathcal{B}(N)) = 2$. This shows that it is not enough to find the bottleneck distance of the barcodes of M and N to decide whether M and N are 1-interleaved and thus whether the CI-problem is solvable. In fact, recent work shows that d_B can be efficiently computed [21].

---

[2] $\mathrm{Hom}(\oplus_i M_i, \oplus_j N_j) \cong \oplus_i \oplus_j \mathrm{Hom}(M_i, N_j)$.

We end this paper with the somewhat surprising observation that the interleaving distance of the above interval decomposable modules depends the characteristic $\mathrm{char}(\mathbb{K})$ of the underlying field $\mathbb{K}$. That is, let $M, N : \mathbf{Z}^2 \to \mathbf{Vec}_{\mathbb{K}}$, $M', N' : \mathbf{Z}^2 \to \mathbf{Vec}_{\mathbb{K}'}$, $\mathbb{K} \neq \mathbb{K}'$, and for which $\mathcal{B}(M) = \mathcal{B}(M')$ and $\mathcal{B}(N) = \mathcal{B}(N')$. Clearly, any matching distance d would satisfy $d(\mathcal{B}(M), \mathcal{B}(N)) = d(\mathcal{B}(M'), \mathcal{B}(N'))$, but it is not always true that $d_I(M, N) = d_I(M', N')$.

For a fixed $n \geq 2$, let $Q = \{(2,2), \dots, (n+2, n+2)\}$ and $P = \{(1,1)\} \cup \{2, \dots, n+2\}^2 \setminus Q$. Then the CI-problem $(P, Q, n+2)$ is solvable if and only if the characteristic of $\mathbb{K}$ divides n. We will only prove this for $n = 2$ for clarity, but the argument easily generalizes to all n.

Assume that (M, M^{-1}) is a solution to $(P, Q, 4)$:

$$M = \begin{bmatrix} 0 & ? & ? & ? \\ a & ? & 0 & 0 \\ b & 0 & ? & 0 \\ c & 0 & 0 & ? \end{bmatrix}, \quad M^{-1} = \begin{bmatrix} ? & d & e & f \\ ? & 0 & ? & ? \\ ? & ? & 0 & ? \\ ? & ? & ? & 0 \end{bmatrix}$$

Here we have put the entries corresponding to the elements of P and Q equal to 0, and left the rest as unknown. The entries we will use in the calculations that follow are labeled a, b, c, d, e, f. We see that $(MM^{-1})_{(2,2)} = ad$, $(MM^{-1})_{(3,3)} = be$, $(MM^{-1})_{(4,4)} = cf$, that is, $ad = be = cf = 1$. At the same time, $(M^{-1}M)_{(1,1)} = ad + be + cf$, so we get $1 = 1+1+1$, or $2 = 0$. Thus $\mathrm{char}(\mathbb{K}) = 2$, and in this case we can put all the unknowns in M and M^{-1} above equal to 1 to obtain a solution. (For $n > 2$, we put the nonzero elements on the diagonal of M equal to -1.)

Our motivation for introducing CI-problems was working towards determining the computational complexity of calculating the interleaving distance. While the last examples say little about complexity, they illustrate the underlying philosophy of our approach: By considering CI-problems, we can avoid the confusing geometric aspects of persistence modules and interleavings. E.g., in the case above, working with persistence modules over a 23×23 size grid is reduced to looking at a pair of 4×4-matrices.

5 Discussion

The problem of determining the computational complexity of computing the interleaving distance for multidimensional persistence modules (valued in **Vec**) was first brought up in Lesnick's thesis [23]. Although it has been an important open question for several years, a non-trivial lower bound on the complexity class has not yet been given. In light of Theorem 2, one might hope that tools from computational algebra can be efficiently extended to the setting of interleavings. Theorem 9 is an argument against this, as it shows that the problem of computing the interleaving distance is NP-hard in general. This leads us to conjecture that the problem of computing the interleaving distance for multidimensional persistence modules is also NP-hard. Unfortunately, writing down the conditions for an interleaving becomes intractable already for small grids. To make the decision problem more accessible to researchers in other fields of mathematics and computer science, we have shown that the problem is at least as hard as an easy to state matrix invertibility problem. We speculate that this problem is also NP-hard. If that is not the case, then an algorithm would provide valuable insight into the interleaving problem for interval decomposable modules.

References

1 Pankaj K Agarwal, Kyle Fox, Abhinandan Nath, Anastasios Sidiropoulos, and Yusu Wang. Computing the gromov-hausdorff distance for metric trees. In *International Symposium on*

Algorithms and Computation, pages 529–540. Springer, 2015.

2 Gorô Azumaya. Corrections and supplementaries to my paper concerning Krull–Remak–Schmidt's theorem. *Nagoya Mathematical Journal*, 1:117–124, 1950.

3 Michael Barot. *Introduction to the representation theory of algebras*. Springer, 2014.

4 Ulrich Bauer and Michael Lesnick. Induced matchings and the algebraic stability of persistence barcodes. *Journal of Computational Geometry*, 6(2):162–191, 2015.

5 Silvia Biasotti, Andrea Cerri, Patrizio Frosini, Daniela Giorgi, and Claudia Landi. Multidimensional size functions for shape comparison. *Journal of Mathematical Imaging and Vision*, 32(2):161–179, 2008.

6 Håvard Bakke Bjerkevik. Stability of higher-dimensional interval decomposable persistence modules. *arXiv preprint arXiv:1609.02086*, 2016.

7 Håvard Bakke Bjerkevik and Magnus Bakke Botnan. Computational complexity of the interleaving distance. *arXiv preprint arXiv:1712.04281*, 2017.

8 Magnus Bakke Botnan. Interval decomposition of infinite zigzag persistence modules. *Proceedings of the American Mathematical Society*, 145(8):3571—-3577, 2017.

9 Magnus Bakke Botnan and Michael Lesnick. Algebraic stability of zigzag persistence modules. *arXiv preprint arXiv:1604.00655*, 2016.

10 Peter A Brooksbank and Eugene M Luks. Testing isomorphism of modules. *Journal of Algebra*, 320(11):4020–4029, 2008.

11 Peter Bubenik, Vin de Silva, and Vidit Nanda. Higher interpolation and extension for persistence modules. *SIAM Journal on Applied Algebra and Geometry*, 1(1):272–284, 2017.

12 Peter Bubenik, Vin de Silva, and Jonathan Scott. Metrics for generalized persistence modules. *Foundations of Computational Mathematics*, 15(6):1501–1531, 2015.

13 Peter Bubenik and Jonathan A Scott. Categorification of persistent homology. *Discrete & Computational Geometry*, 51(3):600–627, 2014.

14 G. Carlsson and A. Zomorodian. The theory of multidimensional persistence. *Discrete & Computational Geometry*, 42(1):71–93, 2009.

15 Gunnar Carlsson and Vin de Silva. Zigzag persistence. *Foundations of computational mathematics*, 10(4):367–405, 2010.

16 Gunnar Carlsson, Vin de Silva, and Dmitriy Morozov. Zigzag persistent homology and real-valued functions. In *Proceedings of the twenty-fifth annual symposium on Computational geometry*, pages 247–256. ACM, 2009.

17 Frédéric Chazal, David Cohen-Steiner, Marc Glisse, Leonidas J Guibas, and Steve Y Oudot. Proximity of persistence modules and their diagrams. In *Proceedings of the twenty-fifth annual symposium on Computational geometry*, pages 237–246. ACM, 2009.

18 Jérémy Cochoy and Steve Oudot. Decomposition of exact pfd persistence bimodules. *arXiv preprint arXiv:1605.09726*, 2016.

19 William Crawley-Boevey. Decomposition of pointwise finite-dimensional persistence modules. *Journal of Algebra and Its Applications*, 14(05):1550066, 2015.

20 Vin de Silva, Elizabeth Munch, and Amit Patel. Categorified Reeb graphs. *arXiv preprint arXiv:1501.04147*, 2015.

21 Tamal K Dey and Cheng Xin. Computing bottleneck distance for 2-d interval decomposable modules. *arXiv preprint arXiv:1803.02869*, 2018.

22 Michael Kerber, Dmitriy Morozov, and Arnur Nigmetov. Geometry helps to compare persistence diagrams. *Journal of Experimental Algorithmics (JEA)*, 22(1):1–4, 2017.

23 Michael Lesnick. Multidimensional interleavings and applications to topological inference. *arXiv preprint arXiv:1206.1365*, 2012.

24 Michael Lesnick. The theory of the interleaving distance on multidimensional persistence modules. *Foundations of Computational Mathematics*, 15(3):613–650, 2015.

25 Nikola Milosavljević, Dmitriy Morozov, and Primoz Skraba. Zigzag persistent homology in matrix multiplication time. In *Proceedings of the twenty-seventh annual symposium on Computational geometry*, pages 216–225. ACM, 2011.

26 Martina Scolamiero, Wojciech Chachólski, Anders Lundman, Ryan Ramanujam, and Sebastian Öberg. Multidimensional persistence and noise. *Foundations of Computational Mathematics*, 17(6):1367–1406, 2017.

27 Cary Webb. Decomposition of graded modules. *Proceedings of the American Mathematical Society*, 94(4):565–571, 1985.

Sheaf-Theoretic Stratification Learning

Adam Brown[1]
Department of Mathematics, University of Utah
Salt Lake City, UT, USA
abrown@math.utah.edu
https://orcid.org/0000-0002-0955-0119

Bei Wang[2]
School of Computing, Scientific Computing and Imaging Institute, University of Utah
Salt Lake City, UT, USA
beiwang@sci.utah.edu
https://orcid.org/0000-0002-9240-0700

Abstract

In this paper, we investigate a sheaf-theoretic interpretation of stratification learning. Motivated by the work of Alexandroff (1937) and McCord (1978), we aim to redirect efforts in the computational topology of triangulated compact polyhedra to the much more computable realm of sheaves on partially ordered sets. Our main result is the construction of stratification learning algorithms framed in terms of a sheaf on a partially ordered set with the Alexandroff topology. We prove that the resulting decomposition is the unique minimal stratification for which the strata are homogeneous and the given sheaf is constructible. In particular, when we choose to work with the local homology sheaf, our algorithm gives an alternative to the local homology transfer algorithm given in Bendich et al. (2012), and the cohomology stratification algorithm given in Nanda (2017). We envision that our sheaf-theoretic algorithm could give rise to a larger class of stratification beyond homology-based stratification. This approach also points toward future applications of sheaf theory in the study of topological data analysis by illustrating the utility of the language of sheaf theory in generalizing existing algorithms.

2012 ACM Subject Classification Mathematics of computing → Algebraic topology, Computing methodologies → Dimensionality reduction and manifold learning

Keywords and phrases Sheaf theory, stratification learning, topological data analysis, stratification

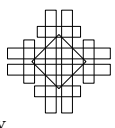
Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.14

Related Version A full version is available at `https://arxiv.org/abs/1712.07734`.

1 Introduction

Our work is motivated by the following question: Given potentially high-dimensional point cloud samples, can we infer the structures of the underlying data? In the classic setting of *manifold learning*, we often assume the support for the data is from a low-dimensional space with manifold structure. However, in practice, a significant amount of interesting data contains mixed dimensionality and singularities. To deal with this more general scenario, we assume the data are sampled from a mixture of possibly intersecting manifolds; the objective is to recover the different pieces, often treated as clusters, of the data associated with different

[1] NSF IIS-1513616
[2] NSF IIS-1513616 and NSF ABI-1661375



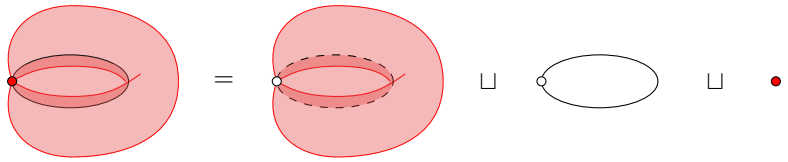


Figure 1 Example of a topological stratification of a *pinched torus*.

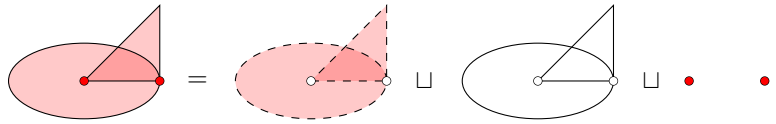


Figure 2 Example of a homological stratification of a *sundial*.

manifolds of varying dimensions. Such an objective gives rise to a problem of particular interest in the field of *stratification learning*. Here, we use the word "stratification learning" loosely to mean an unsupervised, exploratory, clustering process that infers a decomposition of data into disjoint subsets that capture recognizable and meaningful structural information.

Previous work in mathematics has focused on the study of stratified spaces under smooth and continuous settings [12, 25] without computational considerations of noisy and discrete datasets. Statistical approaches that rely on inferences of mixture models and local dimension estimation require strict geometric assumptions such as linearity [13, 16, 24], and may not handle general scenarios with complex singularities. Recently, approaches from topological data analysis [3, 5, 23], which rely heavily on ingredients from computational [9] and intersection homology [2, 4, 10], are gaining momentum in stratification learning.

Topological approaches transform the smooth and continuous setting favored by topologists to the noisy and discrete setting familiar to computational topologists in practice. In particular, the local structure of a point cloud (sampled from a stratified space) can be described by a multi-scale notion of local homology [3]; and the point cloud data could be clustered based on how the local homology of nearby sampled points map into one another [5]. Philosophically, our main goal is to find a stratification where two points in the same strata (or cluster) can not be distinguished by homological methods, and two points in different strata (different clusters) can be distinguished by homological methods. The majority of the paper will be spent developing a rigorous and computable interpretation of the purposely vague statement "distinguished by homological methods". Furthermore, we will see that our approach to computing the above stratification applies equally well to sheaves other than those based on local homology. As our work is an interplay between sheaf theory and stratification, we briefly review various notions of stratification before describing our results.

1.1 Stratifications

Given a topological space X, a *topological stratification* of X is a finite filtration, that is, an increasing sequence of closed subspaces $\emptyset = X_{-1} \subset X_0 \subset \cdots \subset X_d = X$, such that for each i, $X_i - X_{i-1}$ is a (possibly empty) open i-dimensional topological manifold. See Figure 1 for an example of a *pinched torus*, that is, a torus with points along a geodesic with fixed longitude identified, and a spanning disc glued along the equator.

Ideally, we would like to compute a topological stratification for a given space. However, if we are restricted to using only homological methods, this is a dubious task. Topological invariants like homology are too rough to detect when a space such as $X_i - X_{i-1}$ is an open

i-manifold. A well known example of a homological manifold which is not a topological manifold can be constructed from a homology 3-sphere with nontrivial fundamental group. The suspension of such a space is a homological manifold, but not a topological manifold, since the links of the suspension points have nontrivial fundamental groups [18, page 326]. In this paper, we will avoid the difficult problem of computing topological stratifications, and instead aim to investigate stratifications which can be computed using homological methods. Therefore, we must first consider a definition of stratification which does not rely on topological conditions which are not distinguished by homology. We begin with an extremely loose definition of stratification (Definition 1) which only requires the properties necessary to discuss the constructibility of sheaves (defined in Section 2.2). We will then refine our definition of stratification by placing requirements on the constructibility of certain sheaves (Definition 3).

▶ **Definition 1.** Given a topological space X, a *stratification* $\mathfrak{X}$ of X is a finite filtration of X by closed subsets X_i:

$$\emptyset = X_{-1} \subset X_0 \subset \cdots \subset X_d = X.$$

We refer to the space $X_i - X_{i-1}$ as *stratum*, denoted by S_i, and a connected component of S_i as a *stratum piece*.

Suppose we have two stratifications of the topological space X, denoted $\mathfrak{X}$ and $\mathfrak{X}'$. We say that $\mathfrak{X}$ is equivalent to $\mathfrak{X}'$ if each stratum piece of $\mathfrak{X}$ is equal to a stratum piece of $\mathfrak{X}'$.

▶ **Definition 2.** Given two inequivalent stratifications of X, $\mathfrak{X}$ and $\mathfrak{X}'$, we say $\mathfrak{X}$ is *coarser* than $\mathfrak{X}'$ if each stratum piece of $\mathfrak{X}'$ is contained in a stratum piece of $\mathfrak{X}$.

Homological stratification. There have been several approaches in the topology literature to define homological stratifications. While proving the topological invariance of intersection homology, Goresky and MacPherson defined a type of homological stratification which they call a $\bar{p}$-stratification [11, Section 4]. There have been several approaches for building on the ideas of Goresky and MacPherson, with applications to computational geometry and topology in mind ([4], [20]). In this paper, we choose to adopt the perspective of homological stratifications found in [21], with a view toward sheaf theoretic generalizations and applications in topological data analysis.

Consider a filtration $\emptyset = X_{-1} \subset X_0 \subset \cdots \subset X_n = X$ of a topological space X. Let $\mathcal{L}_i$ denote the local homology sheaf on the space X_i (see Section 5.1 for the definition of the local homology sheaf). We say that a stratification is a *homological stratification* if $\mathcal{L}_n$ is locally constant (see Section 2.2 for the definition of locally constant) when restricted to $X_i - X_{i-1}$, for each i. A stratification is a *strong homological stratification* if for each i both $\mathcal{L}_i$ and $\mathcal{L}_n$ are locally constant when restricted to $X_i - X_{i-1}$. Finally, a stratification is a *very strong homological stratification* if for each i and every $k \geq i$, $\mathcal{L}_k$ is locally constant when restricted to $X_i - X_{i-1}$. As mentioned in [21], it would be interesting to study the relationship between these definitions of homological stratification. We plan to pursue this in future work. The cohomological stratification given in [20] can be considered as a cohomological anologue of the very strong homological stratification defined above. The utility of Nanda's definition is the extent to which it lends itself to the study of topological properties of individual strata. For example, it can be easily shown that the strata of such a stratification are R-(co)homology manifolds (R being the ring with which the local cohomology is computed). The trade off for using the very strong homological stratification is in the number of local (co)homology

groups which need to be computed. This is by far the most computationally expensive aspect of the algorithm, and the very strong homological stratification requires one to compute new homology groups for each sheaf $\mathcal{L}_i$. By contrast, our homological stratification only requires the computation of local homology groups corresponding to the sheaf $\mathcal{L}_n$. In this paper, we choose to study the less rigid (and more computable) notion of homological stratification (see Section 4 for more details on computing homological stratifications).

Sheaf-theoretic stratification. The definition of homological stratification naturally lends itself to generalizations, which we now introduce (while delaying formal definition of constructible sheaves to Section 2.2).

▶ **Definition 3.** Suppose $\mathcal{F}$ is a sheaf on a topological space X. An *$\mathcal{F}$-stratification* ("sheaf-stratification") of X is a stratification such that $\mathcal{F}$ is constructible with respect to $X = \amalg S_i$. A *coarsest $\mathcal{F}$-stratification* is an $\mathcal{F}$-stratification such that $\mathcal{F}$ is not constructible with respect to any coarser stratification.

For general topological spaces, a coarsest $\mathcal{F}$-stratification may not exist, and may not be unique if it does exist. The main focus of this paper will be proving existence and uniqueness results for certain coarsest $\mathcal{F}$-stratifications.

1.2 Our contribution

In this paper, we study stratification learning using the tool of constructible sheaves. As a sheaf is designed to systematically track locally defined data attached to the open sets of a topological space, it seems to be a natural tool in the study of stratification based on local structure of the data. Our contributions are three-fold:

1. We prove the existence of coarsest $\mathcal{F}$-stratifications and the existence and uniqueness of the minimal homogeneous $\mathcal{F}$-stratification for finite T_0-spaces (Section 3).
2. We give an algorithm for computing each of the above stratification of a finite T_0-space based on a sheaf-theoretic language (Section 4).
3. In particular, when applying the local homology sheaf in our algorithm, we obtain a coarsest homological stratification (Section 5.2).

In addition, we envision that our abstraction could give rise to a larger class of stratification beyond homological stratification. For instance, we give an example of a "maximal element-stratification" when the sheaf is defined by considering maximal elements of an open set (see the full version [6]).

Comparison to prior work. This paper can be viewed as a continuation of previous works that aim to adapt the stratification and homology theory of Goresky and MacPherson to the realm of topological data analysis. In [21], Rourke and Sanderson give a proof of the topological invariance of intersection homology on PL homology stratifications, and give an recursive process for identifying a homological stratification (defined in Section 5 of [21]). In [4], Bendich and Harer introduce a persistent version of intersection homology that can be applied to simplicial complexes. In [5], Bendich, Wang, and Mukherjee provide computational approach that yields a stratification of point clouds by computing transfer maps between local homology groups of an open covering of the point cloud. In [20], Nanda uses the machinery of derived categories to study cohomological stratifications based on local cohomology.

Motivated by the results of [20] and [5], we aim to develop a computational approach to the stratifications studied in [21]. Our main results can be summarized as the generalization of homological stratifications of [21] to $\mathcal{F}$-stratifications, and a proof of existence and uniqueness

of the minimal homogeneous $\mathcal{F}$-stratification of a finite simplicial complex. When $\mathcal{F}$ is the local homology sheaf, we recover the homological stratification described by [21]. While admitting a similar flavor as [20], our work differs from [20] in several important ways. The most obvious difference is our choice to work with homology and sheaves rather than cohomology and cosheaves. More importantly however, we reduce the number of local homology groups which need to be computed. We will investigate the differences between homological, strong homological, and very strong homological stratifications in future work. Additionally, we plan to build on the results of [5] and [23], and extend the sheaf-theoretic stratification learning perspective described in this paper to the study of stratifications of point cloud data using persistent local homology.

2 Preliminaries

2.1 Compact polyhedra, finite T_0-spaces and posets

Our broader aim is to compute a clustering of a finite set of points sampled from a compact polyhedron, based on the coarsest $\mathcal{F}$-stratification of a finite T_0-space built from the point set. In this paper, we avoid discussion of sampling theory, and assume the finite point set forms the vertex set of a triangulated compact polyhedron. The finite T_0-space is the set of simplices of the triangulation, with the corresponding partial order. To describe this correspondence in more detail, we first consider the connection between compact polyhedra and finite simplicial complexes. We then consider the correspondence between simplicial complexes and T_0-topological spaces.

Compact polyhedra and triangulations. A *compact polyhedron* is a topological space which is homeomorphic to a finite simplicial complex. A *triangulation* of a compact polyhedron is a finite simplicial complex K and a homeomorphism from K to the polyhedron.

T_0-spaces. A T_0*-space* is a topological space such that for each pair of distinct points, there exists an open set containing one but not the other. Its correspondence with simplicial complex is detailed in [17]:

1. For each finite T_0-space X there exists a (finite) simplicial complex K and a weak homotopy equivalence $f : |K| \to X$.
2. For each finite simplicial complex K there exists a finite T_0-space X and a weak homotopy equivalence $f : |K| \to X$.

Here, weak homotopy equivalence is a continuous map which induces isomorphisms on all homotopy groups.

T_0-spaces have a natural partial order. In this paper, we study certain topological properties of a compact polyhedron by considering its corresponding finite T_0-space. The last ingredient, developed in [1], is a natural partial order defined on a given finite T_0-space. We can define this partial ordering on a finite T_0-space X by considering minimal open neighborhoods of each point (i.e. element) $x \in X$. Let X be a finite T_0-space. Each point $x \in X$ has a minimal open neighborhood, denoted B_x, which is equal to the intersection of all open sets containing x.

$$B_x = \bigcap_{U \in \mathcal{N}_x} U,$$

where $\mathcal{N}_x$ denotes the set of open sets containing x. Since X is a finite space, there are only finitely many open sets. In particular, $\mathcal{N}_x$ is a finite set. So B_x is defined to be the

intersection of finitely many open sets, which implies that B_x is an open neighborhood of x. Moreover, any other open neighborhood V of x must contain B_x as a subset. We can define the partial ordering on X by setting $x \leq y$ if $B_y \subseteq B_x$.

Conversely, we can endow any poset X with the Alexandroff topology as follows. For each element $\tau \in X$, we define a minimal open neighborhood containing τ by $B_\tau := \{\gamma \in X : \gamma \geq \tau\}$. The collection of minimal open neighborhoods for each $\tau \in X$ forms a basis for a topology on X. We call this topology the Alexandroff topology. Moreover, a finite T_0-space X is naturally equal (as topological spaces) to X viewed as a poset with the Alexandroff topology. Therefore, we see that each partially ordered set is naturally a T_0-space, and each finite T_0-space is naturally a partially ordered set. The purpose for reviewing this correspondence here is to give the abstractly defined finite T_0-spaces a concrete and familiar realization.

Given a finite T_0-space X with the above partial order, we say $x_0 \leq x_1 \leq \cdots \leq x_n$ (where $x_i \in X$) is a maximal chain in X if there is no totally ordered subset $Y \subset X$ consisting of elements $y_j \in Y$ such that $y_0 \leq \cdots \leq y_j \leq \cdots \leq y_k$ and $\cup_{i=0}^{n}\{x_i\} \subsetneq Y$. The cardinality of a chain $x_0 \leq x_1 \leq \cdots \leq x_n$ is $n+1$. We say that a finite T_0-space has dimension m if the maximal cardinality of maximal chains is $m+1$. An m-dimensional simplicial complex is called *homogeneous* if each simplex of dimension less than m is a face of a simplex of dimension m. Motivated by this definition and the correspondence between simplicial complexes and T_0-spaces, we say an m-dimensional finite T_0-space is *homogeneous* if each maximal chain has cardinality $m+1$.

The correspondences allow us to study certain topological properties of compact polyhedra by using the combinatorial theory of partially ordered sets. In particular, instead of using the more complicated theory of sheaves on the geometric realization $|K|$ of a simplicial complex K, we will continue by studying sheaves on the corresponding finite T_0-space, denoted by X.

2.2 Constructible sheaves

Intuitively, a sheaf assigns some piece of data to each open set in a topological space X, in a way that allows us to glue together data to recover some information about the larger space. This process can be described as the mathematics behind understanding global structure by studying local properties of a space. In this paper, we are primarily interested in sheaves on finite T_0-spaces, which are closely related to the cellular sheaves studied in [22], [8], and [20].

Sheaves. Suppose X is a topological space. Let $\mathbf{Top}(X)$ denote the category consisting of objects which are open sets in X with morphisms given by inclusion. Let $\mathcal{F}$ be a contravariant functor from $\mathbf{Top}(X)$ to $\mathcal{S}$, the category of sets. For open sets $U \subset V$ in X, we refer to the morphism $\mathcal{F}(U \subset V) : \mathcal{F}(V) \to \mathcal{F}(U)$ induced by $\mathcal{F}$ and the inclusion $U \subset V$, as a *restriction* map from V to U. We say that $\mathcal{F}$ is a *sheaf* [3] on X if $\mathcal{F}$ satisfies the following conditions 1-4; a *presheaf* is a functor $\mathcal{E}$ (as above) which satisfies conditions 1-3:

1. $\mathcal{F}(\emptyset) = 0$;
2. $\mathcal{F}(U \subset U) = \mathrm{id}_U$.
3. If $U \subset V \subset W$, then $\mathcal{F}(U \subset W) = \mathcal{F}(U \subset V) \circ \mathcal{F}(V \subset W)$.
4. If $\{V_i\}$ is an open cover of U, and $s_i \in \mathcal{F}(V_i)$ has the property that $\forall i, j$, $\mathcal{F}((V_i \cap V_j) \subset V_i)(s_i) = \mathcal{F}((V_j \cap V_i) \subset V_j)(s_j)$, then there exists a unique $s \in \mathcal{F}(U)$ such that $\forall i$, $\mathcal{F}(V_i \subset U)(s) = s_i$.

[3] See [8, Chapter 2], [15, Chapter 3] for various introductions to sheaf theory.

There is a useful process known as *sheafification*, which allows us to transform any presheaf into a sheaf. In the setting of finite T_0-spaces, sheafification takes on a relatively simple form. Let $\mathcal{E}$ be a presheaf on a finite T_0-space X. Then the sheafification of $\mathcal{E}$, denoted $\mathcal{E}^+$, is given by

$$\mathcal{E}^+(U) = \left\{ f : U \to \coprod_{x \in U} \mathcal{E}(B_x) \middle| f(x) \in \mathcal{E}(B_x) \text{ and } f(y) = \mathcal{F}(B_y \subset B_x)(f(x)) \text{ for all } y \geq x \right\}$$

For any presheaf $\mathcal{E}$, it can be seen that $\mathcal{E}^+$ is necessarily a sheaf. We only need to know the values $\mathcal{E}(B_x)$ for minimal open neighborhoods B_x, and the corresponding restriction maps between minimal open neighborhoods $\mathcal{E}(B_x \subset B_y)$, in order to define the sheafification of $\mathcal{E}$. The result is that two presheaves will sheafify to the same sheaf if they agree on all minimal open neighborhoods. We will use this fact several times in Section 3. Unless otherwise specified, for the remaining of this paper, we use X to denote a T_0-space.

Pull back of a sheaf. For notational convenience, define for each subset $Y \subset X$ the *star* of Y by $\mathrm{St}(Y) := \cup_{y \in Y} B_y$, where B_y is the minimal open neighborhood of $y \in X$. We can think of the star of Y as the smallest open set containing Y. Let X and Y be two finite T_0-spaces. The following property can be thought of as a way to transfer a sheaf on Y to a sheaf on X through a continuous map $f : X \to Y$. Let $\mathcal{F}$ be a sheaf on Y. Then the *pull back* of $\mathcal{F}$, denoted $f^{-1}\mathcal{F}$, is defined to be the sheafification of the presheaf $\mathcal{E}$ which maps an open set $U \subset X$ to $\mathcal{E}(U) := \mathcal{F}(\mathrm{St}(f(U)))$. We can avoid using direct limits in our definition of pull back because each point in a finite T_0-space has a minimal open neighborhood [8, Chapter 5]. The pull back of $\mathcal{F}$ along an inclusion map $\iota : U \hookrightarrow X$ is called the *restriction* of $\mathcal{F}$ to U, and is denoted $\mathcal{F}|_U$.

Constant and locally constant sheaves. Now we can define classes of well-behaved sheaves, constant and locally constant ones, which we can think of intuitively as analogues of constant functions based on definitions common to algebraic geometry and topology [15]. A sheaf $\mathcal{F}$ is a *constant sheaf* if $\mathcal{F}$ is isomorphic to the pull back of a sheaf $\mathcal{G}$ on a single point space $\{x\}$, along the projection map $p : X \to x$. A sheaf $\mathcal{F}$ is *locally constant* if for all $x \in X$, there is a neighborhood U of x such that $\mathcal{F}|_U$ (the restriction of $\mathcal{F}$ to U), is a constant sheaf.

▶ **Definition 4.** A sheaf $\mathcal{F}$ on a finite T_0-space X is *constructible* with respect to the decomposition $X = \coprod S_i$ of X into finitely many disjoint locally closed subsets, if $\mathcal{F}|_{S_i}$ is locally constant for each i.

3 Main results

In this section we state three of our main results, namely, the existence of $\mathcal{F}$-stratifications (Proposition 5), the existence of coarsest $\mathcal{F}$-stratifications (Theorem 6), and the existence and uniqueness of minimal homogeneous $\mathcal{F}$-stratifications (Theorem 9). Of course, Theorem 6 immediately implies Proposition 5. We choose to include a separate statement of Proposition 5 however, as we wish to illustrate the existence of $\mathcal{F}$-stratifications which are not necessarily the coarsest. We include proof sketches here and refer to the full version for technical details.

▶ **Proposition 5.** *Let $\mathcal{F}$ be a sheaf on a finite T_0-space X. There exists an $\mathcal{F}$-stratification of X (see Definition 3 and Definition 4).*

Proof sketch. $\mathcal{F}$ is constructible with respect to the decomposition $X = \coprod_{x \in X} x$. ◀

▶ **Theorem 6.** *Let $\mathcal{F}$ be a sheaf on a finite T_0-space X. There exists a coarsest $\mathcal{F}$-stratification of X.*

Proof sketch. We can prove Theorem 6 easily as follows. There are only finitely many stratifications of our space X, which implies that there must be an $\mathcal{F}$-stratification with a minimal number of strata pieces. Such a stratification must be a coarsest stratification, since any coarser stratification would have fewer strata pieces.

However, the above proof is rather unenlightening if we are interested in computing the coarsest $\mathcal{F}$-stratification. Therefore we include a constructive proof of the existence of a coarsest $\mathcal{F}$-stratification which we sketch here. We can proceed iteratively, by defining the top-dimensional stratum to be the collection of points (i.e. elements) so that the sheaf is constant when restricted to the minimal open neighborhoods of the said points. We then remove the top-dimensional stratum from our space, and pull back the sheaf to the remaining points. We proceed until all the points in our space have been assigned to a stratum. We can see that this is a coarsest $\mathcal{F}$-stratification by arguing that this algorithm, in some sense, maximizes the size of each stratum piece, and thus any coarser $\mathcal{F}$-stratification is actually equivalent to the one constructed above. We refer the reader to the full version for the details of the above argument. ◀

To uniquely identify a stratification by its properties, we will need to introduce a minimal homogeneous $\mathcal{F}$-stratification.

▶ **Definition 7.** Suppose $\mathcal{F}$ is a sheaf on a finite T_0-space X. A *homogeneous $\mathcal{F}$-stratification* is an $\mathcal{F}$-stratification such that for each i, the closure of the stratum S_i in X_i is homogeneous of dimension i (defined in Section 2.1).

We will introduce a lexicographical preorder on the set of homogeneous $\mathcal{F}$-stratifications of a finite T_0-space X. Let $\mathfrak{X}$ be a homogeneous $\mathcal{F}$-stratification of X given by

$$\emptyset = X_{-1} \subset X_0 \subset X_1 \subset \cdots \subset X_n = X$$

We define a sequence $A^{\mathfrak{X}} := \{|X_n|, \cdots, |X_i|, \cdots, |X_0|\}$, where $|X_i|$ denotes the cardinality of the set X_i. Given two stratifications $\mathfrak{X}$ and $\mathfrak{X}'$, we say that $\mathfrak{X}' > \mathfrak{X}$ if the first non-zero term of the sequence $A^{\mathfrak{X}'} - A^{\mathfrak{X}} = \{|X_i'| - |X_i|\}$ is positive; $\mathfrak{X} \leq \mathfrak{X}'$ if there are no non-zero terms. Notice that if $\mathfrak{X}$ and $\mathfrak{X}'$ are homogeneous stratifications such that $\mathfrak{X}$ is coarser than $\mathfrak{X}'$, then we necessarily have that $\mathfrak{X} \leq \mathfrak{X}'$. We say that a stratification $\mathfrak{X}$ is a *minimal homogeneous $\mathcal{F}$-stratification* if $\mathfrak{X} \leq \mathfrak{X}'$ for every other homogeneous $\mathcal{F}$-stratification $\mathfrak{X}'$.

▶ **Definition 8.** A homogeneous $\mathcal{F}$-stratification is *minimal* if it is minimal with respect to the lexocographic order on homogeneous $\mathcal{F}$-stratifications.

There are several examples which illustrate the necessity of introducing *minimal* homogeneous $\mathcal{F}$-stratifications, rather than studying only coarsest homogeneous $\mathcal{F}$-stratifications. Consider the simplicial complex K illustrated in Figure 3. Let $\mathcal{C}$ denote the constant sheaf on the corresponding T_0-space X by assigning the one dimensional vector space k to the star of each simplex σ, $\mathcal{C}(\mathrm{St}\sigma) = k$. For the restriction maps, $\mathcal{C}(\mathrm{St}(\tau) \subset \mathrm{St}(\sigma))$ is an isomorphism for each pair $\sigma < \tau$. Figure 3 shows that there are two coarsest homogeneous $\mathcal{C}$-stratifications of X. The stratification on the right side of Figure 3 is the minimal homogeneous $\mathcal{C}$-stratification.

▶ **Theorem 9.** *Let K be a finite simplicial complex, and X be a finite T_0-space consisting of the simplices of K endowed with the Alexandroff topology. Let $\mathcal{F}$ be a sheaf on X. There exists a unique minimal homogeneous $\mathcal{F}$-stratification of X. Moreover, the unique minimal homogeneous $\mathcal{F}$-stratification is a coarsest homogeneous $\mathcal{F}$-stratification.*

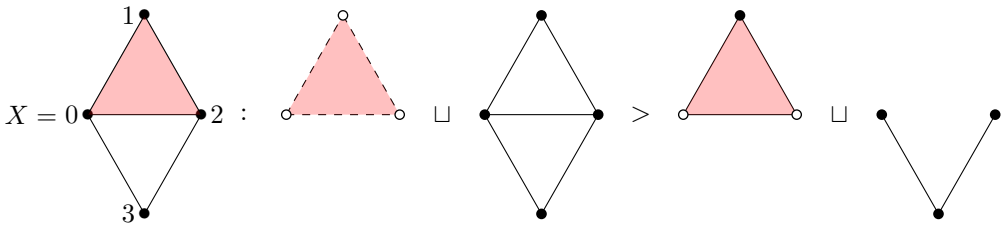


Figure 3 An example of two inequivalent coarsest homogeneous $\mathcal{C}$-stratifications, where $\mathcal{C}$ is a constant sheaf on X. The stratification given on the right is the minimal homogeneous $\mathcal{C}$-stratification.

Proof sketch. The idea for this proof is very similar to that of the Theorem 6. We construct a stratification in a very similar way, with the only difference being that we must be careful to only construct homogeneous strata. The argument for the uniqueness of the resulting stratification uses the observation that this iterative process maximizes the size of the current stratum (starting with the top-dimensional stratum) before moving on to define lower-dimensional strata. Thus the resulting stratification is minimal in the lexocographic order. The top-dimensional stratum of any other minimal homogeneous $\mathcal{F}$-stratification then must equal the top stratum constructed above, since these must both include the set of top-dimensional simplices, and have maximal size. An inductive argument then shows the stratifications are equivalent. The comments preceding Definition 8 imply that the minimal homogeneous $\mathcal{F}$-stratification is a coarsest homogeneous $\mathcal{F}$-stratification. Again, we refer readers to the full version for the remaining details. ◀

4 A sheaf-theoretic stratification learning algorithm

We outline an explicit algorithm for computing a coarsest $\mathcal{F}$-stratification of a space X given a particular sheaf $\mathcal{F}$. We give two examples of stratification learning using the local homology sheaf (Section 5) and the sheaf of maximal elements in the full version.

Let X be a finite T_0-space, equipped with a partial ordering. Instead of using the sheaf-theoretic language of Theorem 9, we frame the computation in terms of X and an "indicator function" δ. For every $x, y \in X$ with a relation $x \leq y$, δ assigns a binary value to the relation. That is, $\delta(x \leq y) = 1$ if the restriction map $\mathcal{F}(B_y \subset B_x) : \mathcal{F}(B_x) \to \mathcal{F}(B_y)$ is an isomorphism, and $\delta(x \leq y) = 0$ otherwise. We say a pair $w \leq y$ is *adjacent* if $w \leq z \leq y$ implies $z = w$ or $z = y$ (in other words, there are no elements in between w and y). Due to condition 3 in the definition of a sheaf (Section 2.2), δ is fully determined by the values $\delta(w \leq y)$ assigned to each adjacent pair (w, y). If $a_1 \leq a_2 \leq \cdots \leq a_k$ is a chain of adjacent elements (a_i is adjacent to a_{i+1} for each i), we have that $\delta(a_1 \leq a_k) = \delta(a_1 \leq a_2) \cdot \delta(a_2 \leq a_3) \cdots \delta(a_{k-1} \leq a_k)$. As X is equipped with a finite partially ordering, computing δ can be interpreted as assigning a binary label to the edges of a Hasse diagram associated with the partial ordering (see Section 5 for an example).

For simplicity, we assume that δ is pre-computed, with a complexity of $O(m)$ where m denotes the number of adjacent relations in X. When X corresponds to a simplicial complex K, m is the number of nonzero terms in the boundary matrices of K. δ can, of course, be processed on-the-fly, which may lead to more efficient algorithm. In addition, determining the value of δ is a local computation for each $x \in X$, therefore it is easily parallelizable.

Computing a coarsest $\mathcal{F}$-stratification. If we are only concerned with calculating a coarsest $\mathcal{F}$-stratification as described in Theorem 6, we may use the algorithm below.

1. Set $i = 0$, $d_0 = \dim X$, $X_{d_0} = X$, and initialize $S_j = \emptyset$, for all $0 \leq j \leq d_0$.
2. While $d_i \geq 0$, do
   a. For each $x \in X_{d_i}$, set $S_{d_i} = S_{d_i} \cup x$ if $\delta(w \leq y) = 1$, $\forall$ adjacent pairs $w \leq y$ in $B_x \cap X_{d_i}$
   b. Set $d_{i+1} = \dim(X_{d_i} - S_{d_i})$
   c. Define $X_{d_{i+1}} = X_{d_i} - S_{d_i}$
   d. Set $i = i + 1$
3. Return S

Here, i is the step counter; d_i is the dimension of the current strata of interest; the set S_{d_i} is the stratum of dimension d_i. d_i decreases from $\dim(X)$ to 0. To include an element x to the current stratum S_{d_i}, we need to check δ for adjacent relations among all x's cofaces.

Computing the unique minimal homogeneous $\mathcal{F}$-stratification. If we would like to obtain the unique minimal homogeneous $\mathcal{F}$-stratification, then we need to modify step 2a. Let $c(x, i) = 1$ if all maximal chains in X_{d_i} containing x have cardinality d_i, and $c(x, i) = 0$ otherwise. Then the modified version of 2.a. is:

2.a. For each $x \in X_{d_i}$, set $S_{d_i} = S_{d_i} \cup x$ if $\delta(w \leq y) = 1, \forall$ adj. pairs $w \leq y$ in $B_x \cap X_{d_i}$ and $c(x, i) = 1$

5 Stratification learning with local homology sheaf

5.1 Local homology sheaf

For a finite T_0-space X, consider the chain complex $C_\bullet(X)$, where $C_p(X)$ denotes a free R-module generated by $(p+1)$-chains in X, with chain maps $\partial_p : C_p(X) \to C_{p-1}(X)$ given by

$$\partial_p(a_0 \leq \cdots \leq a_p) = \sum (-1)^i (a_0 \leq \cdots \leq \hat{a_i} \leq \cdots \leq a_p)$$

where $\hat{a_i}$ means that the element a_i is to be removed from the chain.

We would like to remark on the decision to refer to this sheaf as the local homology sheaf. If X is a more general topological space (CW space, simplicial complex, manifold, etc), then the *local homology* of X at $x \in X$ is defined to be the direct limit of relative homology $H_\bullet(X, X - x) := \varinjlim H_\bullet(X, X - U)$, where the direct limit is taken over all open neighborhoods U of x with the inclusion partial order [19, page 196]. In our setting, the local homology of X (a finite T_0-space) at a point $x \in X$ is given by $H_\bullet(X, X - B_x)$. Here we avoid using notions of direct limit by working with topological spaces that have minimal open neighborhoods. This motivates us to refer to the sheaf defined by relative homology $H_\bullet(X, X - U)$ for each open set U (see Theorem 10), as the *local homology sheaf* [4]. The following theorem, though straightforward, provides justification for applying the results of Section 4 to local homology computations (see the full version for its proof).

▶ **Theorem 10.** *The functor $\mathcal{L}$ from the category of open sets of a finite T_0-space to the category of graded R-modules, defined by*

$$\mathcal{L}(U) := H_\bullet(X, X - U)$$

where R is the ring of coefficients of the relative homology, is a sheaf on X.

[4] See [7] for an interesting approach to the computation of homology groups of finite T_0-spaces.

5.2 An example of stratification learning using local homology sheaf

If X is a T_0-space corresponding to a simplicial complex K, then the local homology groups in Section 5.1 are isomorphic to the simplicial homology groups of K. We now give a detailed example of stratification learning using local homology sheaf for the sundial example from Figure 2. We will abuse notation slightly, and use K to denote the finite T_0-space consisting of elements which are open simplices corresponding to the triangulated sundial (Figure 4). We choose this notation so that we can describe our T_0-space using the more familiar language of simplicial complexes. For a simplex $\sigma \in K$, its minimal open neighborhood B_σ is its *star* consisting of all cofaces of σ, $\mathrm{St}(\sigma) = \{\tau \in K \mid \sigma \leq \tau\}$. The *closed star*, $\overline{\mathrm{St}}(\sigma)$, is the smallest subcomplex that contains the star. The *link* consists of all simplices in the closed star that are disjoint from the star, $\mathrm{Lk}(\sigma) = \{\tau \in \overline{\mathrm{St}}(\sigma) \mid \tau \cap \mathrm{St}(\sigma) = \emptyset\}$. K is equipped with a partial order based on face relations, where $x < y$ if x is a proper face of y. This partial order gives rise to a Hasse diagram illustrated in Figure 5.

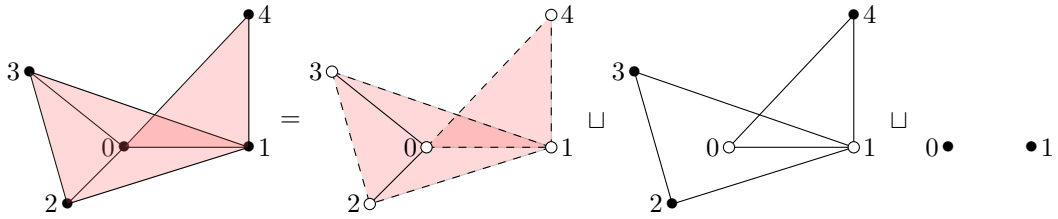


Figure 4 A triangulated sundial and its stratification based on the local homology sheaf.

A sheaf on K can be considered as a labeling of each vertex in the Hasse diagram with a set and each edge with a morphism between the corresponding sets. Consider the local homology sheaf $\mathcal{L}$ on K which takes each open set [5] $U \subset K$ to $H_\bullet(K, K-U) \cong \tilde{H}_\bullet(K/(K-U)) \cong \tilde{H}_\bullet(\mathrm{Cl}(U)/\mathrm{Lk}(U)) \cong H_\bullet(\mathrm{Cl}(U), \mathrm{Lk}(U))$, where $\mathrm{Lk}(U) := \mathrm{Cl}(U) - U$. The above isomorphisms follow from excision and the observation that $K - U$ (resp. $\mathrm{Lk}(U)$) is a closed subcomplex of K (resp. $\mathrm{Cl}(U)$), and therefore $(K, K-U)$ and $(\mathrm{Cl}(U), \mathrm{Lk}(U))$ form good pairs (see [14, page 124]). Our algorithm described in Section 4 can then be interpreted

[5] In the finite simplicial setting, U is the support of a union of open simplices in K.

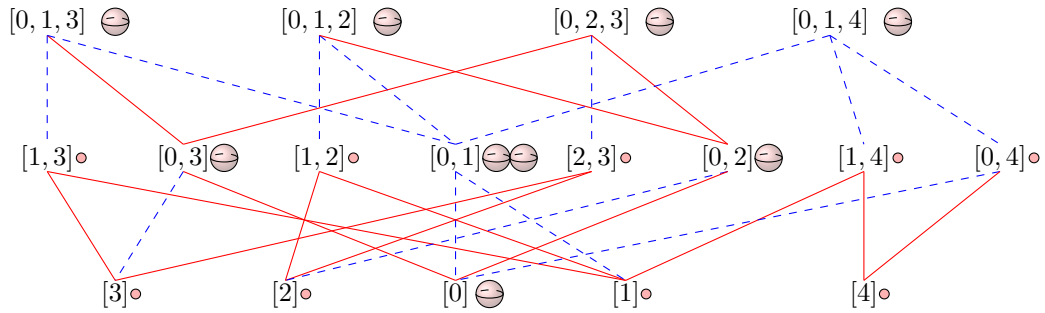


Figure 5 The Hasse diagram of the triangulated sundial. For any two adjacent simplices $\tau < \sigma$, an edge between τ and σ in the diagram is solid red if $\mathcal{L}(B_\tau) \to \mathcal{L}(B_\sigma)$ is an isomorphism; otherwise it is in dotted blue. On the right of each simplex τ is either a point, a sphere, or the wedge of two spheres, chosen so that $\mathcal{L}(B_\tau)$ is isomorphic to the reduced homology of the associated space.

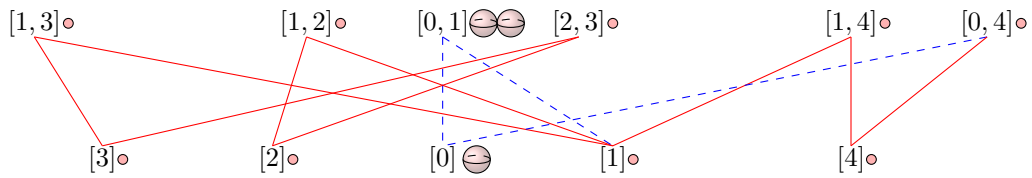


Figure 6 The Hasse diagram after the top dimensional stratum has been removed. We can consider this the beginning of the second iteration of the algorithm in Section 4.

as computing local homology sheaf associated with each vertex in the Hasse diagram, and determining whether each edge in the diagram is an isomorphism. Our algorithm works by considering an element σ in the Hasse diagram to be in the top-dimensional strata if all of the edges above σ are isomorphisms, that is, if $\mathcal{L}(\sigma < \tau)$ is an isomorphism for all pairs $\sigma < \tau$.

As illustrated in Figure 5, first, we start with the 2-simplexes. Automatically, we have that $\mathcal{L}$ is constant when restricted to any 2-simplex, and gives homology groups isomorphic to the reduced homology of a 2-sphere. For instance, the local homology groups of the 2-simplex $\sigma = [0, 1, 3]$ is isomorphic to the reduced homology of a 2-sphere, $H_\bullet(\overline{\mathrm{St}}(\sigma), \mathrm{Lk}(\sigma)) \cong \tilde{H}_\bullet(S^2)$.

Second, we consider the restriction of $\mathcal{L}$ to the minimal open neighborhood of a 1-simplex. For instance, consider the 1-simplex $[1, 3]$; $B_{[1,3]} = [1, 3] \cup [0, 1, 3]$. It can be seen that $\mathrm{Lk}(B)_{[1,3]} = [0] \cup [3] \cup [1] \cup [0, 3] \cup [0, 1]$, and $H_\bullet(\mathrm{Cl}(B_{[1,3]}), \mathrm{Lk}(B_{[1,3]}))$ is isomorphic to the reduced homology of a single point space. Therefore the restriction map $\mathcal{L}(B_{[1,3]}) \to \mathcal{L}(B_{[0,1,3]})$ is not an isomorphism (illustrated as a dotted blue line in Figure 5). On the other hand, let us consider the 1-simplex $[0, 3]$, where $B_{[0,3]} = [0, 3] \cup [0, 1, 3] \cup [0, 2, 3]$. We have that $\mathrm{Lk}(B_{[0,3]}) = [0] \cup [1] \cup [2] \cup [3] \cup [0, 1] \cup [0, 2] \cup [1, 3] \cup [2, 3] \cup [1, 3]$. Therefore $\mathcal{L}(B_{[0,3]})$ is isomorphic to the reduced homology of a 2-sphere. Moreover, both of the restriction maps corresponding to $B_{[0,1,3]} \subset B_{[0,3]}$ and $B_{[0,2,3]} \subset B_{[0,3]}$ are isomorphisms (illustrated as solid red lines in Figure 5). This implies that $[0, 3] \in S_2 = X_2 - X_1$. Alternatively, we can consider the simplex $[0, 1]$ and see that $\mathcal{L}(B_{[0,1]}) \cong \tilde{H}_\bullet(\overline{\mathrm{St}}([0, 1])/\mathrm{Lk}([0, 1])) \cong \tilde{H}_\bullet(S^2 \vee S^2)$, the reduced homology of the wedge of two spheres. Therefore, for any 2-simplex τ, the restriction map $\mathcal{L}([0, 1] < \tau)$ can not be an isomorphism. We conclude that $[0, 1]$ is not contained in the top dimensional stratum. If we continue, we see that the top dimensional stratum is given by $S_2 = [0, 1, 3] \cup [0, 1, 2] \cup [0, 2, 3] \cup [0, 1, 4] \cup [0, 2] \cup [0, 3]$, see Figure 4.

Next, we can calculate the stratum $S_1 = X_1 - X_0$ by only considering restriction maps whose codomain is not contained in S_2 (see Figure 6). We get $S_1 = [0, 1] \cup [1, 3] \cup [1, 2] \cup [2, 3] \cup [0, 4] \cup [1, 4] \cup [2] \cup [3] \cup [4]$, which is visualized in Figure 4. Finally, the stratum $S_0 = X_0$ consists of the vertices which have not been assigned to any strata. So $S_0 = [0] \cup [1]$.

We observe that (for this example) the coarsest $\mathcal{L}$-stratification we calculated is actually the unique minimal homogeneous $\mathcal{L}$-stratification. We will investigate this coincidence for $\mathcal{L}$-stratifications elsewhere, to verify if a coarsest $\mathcal{L}$-stratification is automatically homogeneous or minimal. For low-dimensional examples, we observe that the local homology based stratification we recover is actually a topological stratification. For instance, the coarsest $\mathcal{L}$-stratification we obtain from our algorithm coincides with the stratification given in Figure 1 (for a suitable triangulation of the pinched torus). In general, local homology does not carry enough information to recover a stratification into manifold pieces, and examples exist in higher dimensions where $\mathcal{L}$-stratification are not topological stratifications.

6 Discussion

A triangulation of the pinched torus in Figure 1 can be thought of as a stratification of the space, by defining the d-dimensional stratum to be the collection of d-dimensional simplices. However, this stratification is too "fine" in a sense, as it breaks up the underlying space into too many pieces, resulting in each stratum piece retaining relatively little information about the structure of the total space. The results of this paper can be interpreted as a method for computing a coarsening of the stratification obtained from the triangulation of our underlying space, using sheaf-theoretic techniques (in particular, homological techniques) to determine when two simplices should belong to the same coarser stratum.

There are two key features of the sheaf-theoretic stratification learning algorithm which should be highlighted. The first feature is that we avoid computations which require the sheafification process. At first glance this may be surprising to those not familiar with cellular sheaves, since constructible sheaves can not be defined without referencing sheafification, and our algorithm builds a stratification for which a given sheaf is constructible. In other words, each time we want to determine the restriction of a sheaf to a subspace, we need to compute the sheafification of the presheaf referenced in the definition of the pull back of a sheaf (Section 2.2). We can avoid this by noticing two facts. Suppose $\mathcal{E}$ is a presheaf and $\mathcal{E}^+$ is the sheafification of $\mathcal{E}$. First, in the setting of finite T_0-spaces, we can deduce if $\mathcal{E}^+$ is constant by considering how it behaves on minimal open neighborhoods. Second, the behavior of $\mathcal{E}^+$ will agree with the behavior of the presheaf $\mathcal{E}$ on minimal open neighborhoods. Symbolically, this is represented by the equalities $\mathcal{E}^+(B_x) = \mathcal{E}(B_x)$ and $\mathcal{E}^+(B_w \subset B_x) = \mathcal{E}(B_w \subset B_x)$ for all pairs of minimal open neighborhoods $B_w \subset B_x$ (where B_x is a minimal open neighborhood of x, and B_w is a minimal open neighborhood of an element $w \in B_x$). Therefore, we can determine if $\mathcal{E}^+$ is constant, locally constant, or constructible, while only using computations involving the presheaf $\mathcal{E}$ applied to minimal open neighborhoods.

The second feature of our algorithm (which is made possible by the first) is that the only sheaf-theoretic computation required is checking if $\mathcal{F}(B_w \subset B_x)$ is an isomorphism for each pair $B_w \subset B_x$ in our space. This is extremely relevant for implementations of the algorithm, as it minimizes the number of expensive computations required to build an $\mathcal{F}$-stratification. For example, if our sheaf is the local homology sheaf, we will only need to compute the restriction maps between local homology groups of minimal open neighborhoods. The computation of local homology groups can therefore be distributed and computed independently. Additionally, once we have determined whether the local homology restriction maps are isomorphisms, we can quickly compute a coarsest $\mathcal{F}$-stratification, or a minimal homogeneous $\mathcal{F}$-stratification, without requiring any local homology groups to be recomputed.

There are several interesting questions related to $\mathcal{F}$-stratifications that we will investigate in the future. We are interested in studying $\mathcal{F}$-stratifications for natural sheaves other than the local homology sheaf, on finite T_0-spaces. The primary objective would be to find easily computable sheaves that yield intuitive stratifications relevant for manifold learning algorithms and of interest to data analysts. We will also study the stability of $\mathcal{F}$-stratifications under refinements of triangulations of polyhedra. In this direction, it would be interesting to view $\mathcal{F}$-stratifications from the perspective of persistent homology. If we are given a point cloud sampled from a compact polyhedron, it would be natural to ask about the convergence of $\mathcal{F}$-stratifications and the properties of the strata under a filtration of the simplicial complex. Finally, we are also intrigued by the results of [7], and possible implementations of our algorithm using spectral sequences.

References

1. Pavel Sergeyevich Alexandroff. Diskrete Räume. *Mathematiceskii Sbornik*, 2:501–518, 1937.
2. Paul Bendich. *Analyzing Stratified Spaces Using Persistent Versions of Intersection and Local Homology*. PhD thesis, Duke University, 2008.
3. Paul Bendich, David Cohen-Steiner, Herbert Edelsbrunner, John Harer, and Dmitriy Morozov. Inferring local homology from sampled stratified spaces. *IEEE Symposium on Foundations of Computer Science*, pages 536–546, 2007.
4. Paul Bendich and John Harer. Persistent intersection homology. *Foundations of Computational Mathematics*, 11:305–336, 2011.
5. Paul Bendich, Bei Wang, and Sayan Mukherjee. Local homology transfer and stratification learning. *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1355–1370, 2012.
6. Adam Brown and Bei Wang. Sheaf-theoretic stratification learning. *arXiv:1712.07734*, 2017.
7. Nicolás Cianci and Miguel Ottina. A new spectral sequence for homology of posets. *Topology and its Applications*, 217:1–19, 2017.
8. Justin Curry. *Sheaves, Cosheaves and Applications*. PhD thesis, University of Pennsylvania, 2014.
9. Herbert Edelsbrunner and John Harer. *Computational Topology: An Introduction*. American Mathematical Society, 2010.
10. Mark Goresky and Robert MacPherson. Intersection homology I. *Topology*, 19:135–162, 1982.
11. Mark Goresky and Robert MacPherson. Intersection homology II. *Inventiones Mathematicae*, 71:77–129, 1983.
12. Mark Goresky and Robert MacPherson. *Stratified Morse Theory*. Springer-Verlag, 1988.
13. Gloria Haro, Gregory Randall, and Guillermo Sapiro. Stratification learning: Detecting mixed density and dimensionality in high dimensional point clouds. *Advances in Neural Information Processing Systems (NIPS)*, 17, 2005.
14. Allen Hatcher. *Algebraic Topology*. Cambridge University Press, 2002.
15. Frances Clare Kirwan. *An introduction to intersection homology theory*. Chapman & Hall/CRC, 2006.
16. Gilad Lerman and Teng Zhang. Probabilistic recovery of multiple subspaces in point clouds by geometric lp minimization. *Annals of Statistics*, 39(5):2686–2715, 2010.
17. Michael C. McCord. Singular homology groups and homotopy groups of finite topological spaces. *Duke Mathematical Journal*, 33:465–474, 1978.
18. Washington Mio. Homology manifolds. *Annals of Mathematics Studies (AM-145)*, 1:323–343, 2000.
19. James R. Munkres. *Elements of algebraic topology*. Addison-Wesley, Redwood City, CA, USA, 1984.
20. Vidit Nanda. Local cohomology and stratification. ArXiv:1707.00354, 2017.
21. Colin Rourke and Brian Sanderson. Homology stratifications and intersection homology. *Geometry and Topology Monographs*, 2:455–472, 1999.
22. Allen Dudley Shepard. *A Cellular Description Of The Derived Category Of A Stratified Space*. PhD thesis, Brown University, 1985.
23. Primoz Skraba and Bei Wang. Approximating local homology from samples. *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 174–192, 2014.
24. René Vidal, Yi Ma, and Shankar Sastry. Generalized principal component analysis (GPCA). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:1945–1959, 2005.
25. Shmuel Weinberger. *The topological classification of stratified spaces*. University of Chicago Press, Chicago, IL, 1994.

Realizations of Indecomposable Persistence Modules of Arbitrarily Large Dimension

Mickaël Buchet
Graz University of Technology
Graz, Austria
buchet@tugraz.at

Emerson G. Escolar[1]
WPI-AIMR, Tohoku University
Sendai, Japan
e.g.escolar@gmail.com

Abstract

While persistent homology has taken strides towards becoming a widespread tool for data analysis, multidimensional persistence has proven more difficult to apply. One reason is the serious drawback of no longer having a concise and complete descriptor analogous to the persistence diagrams of the former. We propose a simple algebraic construction to illustrate the existence of infinite families of indecomposable persistence modules over regular grids of sufficient size. On top of providing a constructive proof of representation infinite type, we also provide realizations by topological spaces and Vietoris-Rips filtrations, showing that they can actually appear in real data and are not the product of degeneracies.

2012 ACM Subject Classification Mathematics of computing → Algebraic topology, Theory of computation → Computational geometry

Keywords and phrases persistent homology, multi-persistence, representation theory, quivers, commutative ladders, Vietoris-Rips filtration

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.15

Related Version A full version of this paper is available at `https://arxiv.org/abs/1803.05722`

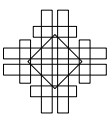
1 Introduction

Recently, persistent homology [8] has established itself as the flagship tool of topological data analysis. It provides the persistence diagrams, an easy to compute and understand compact summary of topological features in various scales in a filtration. Fields where it has been successfully applied include materials science [12, 14], neuroscience [10, 13], genetics [7] or medicine [6, 17].

Over a filtration, persistence diagrams can be used because the persistence module can be uniquely decomposed into indecomposable modules which are intervals. To these intervals, we associate the lifespans of topological features. However, considering persistence modules over more general underlying structures, indecomposables are no longer intervals and can be more complicated.

As an example, in multidimensional persistence [5] over the commutative grid, the representation category is no longer representation finite. In other words, the number of possible indecomposables is infinite for a large enough finite commutative grid. The

[1] E.G.E. is supported by JST CREST Mathematics 15656429.



minimal size for a commutative grid to be representation infinite is relatively small. For two dimensional grids, we need a size of at least 2×5 or 3×3, as the $2 \times n$ grid is representation finite for $n \leq 4$ [9]. When considering three dimensional grids, it is enough to have a $2 \times 2 \times 2$ grid. For a particular finite dimensional persistence module, it is true that it can be uniquely decomposed into a direct sum of indecomposables, but we cannot list all the possible indecomposables a priori.

From another point of view, recent progress in software [15] has made practical application of multidimensional persistence more accessible. However, they approach this by computing incomplete invariants instead of indecomposable decompositions. This has the advantage of being easier to compute than the full decomposition and easier to visualize.

In this work, we aim to provide more intuition for the structure of some indecomposable modules. First, we provide algebraic constructions of some infinite families of indecomposable modules over all representation infinite grids.

Next, we tackle the problem of realizing these constructions. Any module over a commutative grid can realized as the k-th persistent homology module of a simplicial complex for any $k > 0$. This result was first claimed in [5] and proved in [11] with a construction which is straightforward algebraically but difficult to visualize.

Here, we realize our infinite families of indecomposable modules using a more visual construction. In all our algebraic constructions, our infinite families depend on a dimension parameter d and a parameter $\lambda \in K$, which we set to $\lambda = 0$ for our topological constructions. Our topological constructions are formed by putting together d copies of a repeating simple pattern. Finally we provide a Vietoris-Rips construction for our realization of the 2×5 case. Moreover, we show that this construction is stable with respect to small perturbations.

A direct corollary of our result is a constructive proof of the representation infinite type of the grids we consider. It also provides insight into one possible topological origin of these algebraic complications. Given that the construction is stable with respect to noise and appears through a relatively simple configuration, we argue that these kinds of complicated indecomposables may appear when applying multidimensional persistence to real data. Therefore these structures cannot be ignored and we hope that our construction provides insight into a source of indecomposability.

2 Background

We start with a quick overview of the necessary background. First we recall some basic definitions from the representation theory of bound quivers and detail how we check the indecomposability of representations. More details can be found in [2], for example. In the second part, we explain the block matrix formalism [1] we use to simplify some computations. We assume some familiarity with algebraic topology [16], in particular homology.

2.1 Representations of bound quivers

A quiver is a directed graph. In this work, we consider only quivers with a finite number of vertices and arrows and no cycles. A particular example is the linear quiver with n vertices. Let $n \in \mathbb{N}$ and $\tau = (\tau_1, \tau_2, \ldots, \tau_{n-1})$ be a sequence of symbols $\tau_i = f$ or $\tau_i = b$. Below, the two-headed arrow $\longleftrightarrow$ stands for either an arrow pointing to the right or left. The quiver $\mathbb{A}_n(\tau)$ is the quiver with n vertices and $n-1$ arrows, where the i^{th} arrow points to the right if $\tau_i = f$ and to the left otherwise: $\bullet \longleftrightarrow \bullet \longleftrightarrow \ldots \longleftrightarrow \bullet$. In the case that all arrows are pointing forwards, we use the notation $\vec{\mathbb{A}}_n = \mathbb{A}_n(f \ldots f)$.

Throughout this work, let K be a field. A representation V of a quiver Q is a collection $V = (V_i, V_\alpha)$ where V_i is a finite dimensional K-vector space for each vertex i of Q, and each *internal map* $V_\alpha : V_i \to V_j$ is a linear map for each arrow α from i to j in Q.

A homomorphism from V to W, both representations of the same quiver Q, is a collection of linear maps $\{f_i : V_i \to W_i\}$ ranging over vertices i of Q such that $W_\alpha f_i = f_j V_\alpha$ for each arrow α from i to j. The set of all homomorphisms from V to W is the K-vector space $\mathrm{Hom}(V, W)$. The endomorphism ring of a representation V is $\mathrm{End}(V) = \mathrm{Hom}(V, V)$.

General quivers do not impose any constraints on the internal maps of their representations. However, we will mostly consider the case of commutative quivers, a special kind of quiver bound by relations. In particular, representations of a commutative quiver are required to satisfy the property that they form a commutative diagram. For more details see [2]. In the sequel, we shall take Q to mean either a quiver or a commutative quiver, depending on context, and $\mathrm{rep}\, Q$ its category of representations.

The language of representation theory was introduced to persistence in [4], where zigzag persistence modules were considered as representations of $\mathbb{A}_n(\tau)$. From now, we use the term persistence module over Q interchangeably with a representation of Q.

Any pair of representations $V = (V_i, V_\alpha)$ and $W = (W_i, W_\alpha)$ of a quiver Q has a direct sum $V \oplus W = (V_i \oplus W_i, V_\alpha \oplus W_\alpha)$ which is also a representation of Q. A representation V is said to be indecomposable if $V \cong W \oplus W'$ implies that either W or W' is the zero representation. We are concerned with the indecomposability of representations and use the following property relating the endomorphism ring with indecomposability.

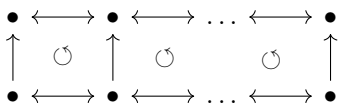
▶ **Definition 1.** Let R be a ring with unity. R is said to be *local* if $0 \neq 1$ in R and for each $x \in R$, x or $1 - x$ is invertible.

▶ **Lemma 2** (Corollary 4.8 of [2]). *Let V be a representation of a (bound) quiver Q.*
1. *If* $\mathrm{End}\, V$ *is local then V is indecomposable.*
2. *If V is finite dimensional and indecomposable, then* $\mathrm{End}\, V$ *is local.*

2.2 Block matrix formalism

Our first construction will be for a particular family of bound quivers called commutative ladders, which are the commutative grids of size $2 \times n$.

▶ **Definition 3.** The commutative ladder of length n with orientation τ, denoted $CL_n(\tau)$ is



which is a quiver with two copies of $\mathbb{A}_n(\tau)$ with the same orientation τ for the top and bottom rows, and bound by all commutativity relations.

Let us review the block matrix formalism for persistence modules on commutative ladders $CL_n(\tau)$ introduced in [1]. We denote by $\mathrm{arr}(\mathrm{rep}\, \mathbb{A}_n(\tau))$ the *arrow category* (also known as the *morphism category*) of $\mathrm{rep}\, \mathbb{A}_n(\tau)$, which is formed by the morphisms of $\mathrm{rep}\, \mathbb{A}_n(\tau)$ as objects. The following proposition allows us to identify representations of the commutative ladder $CL_n(\tau)$ with morphisms between representations of $\mathbb{A}_n(\tau)$. Since the structure of the latter is well-understood, we use this to simplify some computations.

▶ **Lemma 4.** *Let τ be an orientation of length n. There is an isomorphism of K-categories* $F : \mathrm{rep}\, CL_n(\tau) \cong \mathrm{arr}(\mathrm{rep}\, \mathbb{A}_n(\tau))$.

Proof. Given $M \in \operatorname{rep} CL_n(\tau)$, the bottom row (denoted M_1) of M and the top row (denoted M_2) of M are representations of $\mathbb{A}_n(\tau)$. By the commutativity relations imposed on M, the internal maps of M pointing upwards defines a morphism $\phi : M_1 \to M_2$ in $\operatorname{rep} \mathbb{A}_n(\tau)$. The functor F maps M to this morphism and admits an obvious inverse. ◀

Each morphism $(\phi : U \to V) \in \operatorname{arr}(\operatorname{rep} \mathbb{A}_n(\tau))$ has representations of $\mathbb{A}_n(\tau)$ as domain and codomain. As such, they each can be decomposed into interval representations. Thus, ϕ is isomorphic to some

$$\Phi : \bigoplus_{1 \le a \le b \le n} \mathbb{I}[a,b]^{m_{a,b}} \to \bigoplus_{1 \le c \le d \le n} \mathbb{I}[c,d]^{m'_{c,d}}.$$

Relative to these decompositions, Φ can be written in a block matrix form $\Phi = [\Phi^{c:d}_{a:b}]$ where each block is defined by composition with the corresponding inclusion and projection

$$\Phi^{c:d}_{a:b} : \mathbb{I}[a,b]^{m_{a,b}} \xrightarrow{\iota} \bigoplus_{1 \le a \le b \le n} \mathbb{I}[a,b]^{m_{a,b}} \xrightarrow{\Phi} \bigoplus_{1 \le c \le d \le n} \mathbb{I}[c,d]^{m'_{c,d}} \xrightarrow{\pi} \mathbb{I}[c,d]^{m'_{c,d}}.$$

Next, we analyze these blocks by looking at the homomorphism spaces between intervals.

▶ **Definition 5.** The relation $\trianglerighteq$ is the relation on the set of interval representations of $\mathbb{A}_n(\tau)$, $\{\mathbb{I}[b,d] : 1 \le b \le d \le n\}$, such that $\mathbb{I}[a,b] \trianglerighteq \mathbb{I}[c,d]$ if and only if $\operatorname{Hom}(\mathbb{I}[a,b], \mathbb{I}[c,d])$ is nonzero.

▶ **Lemma 6** (Lemma 1 of [1]). *Let $\mathbb{I}[a,b], \mathbb{I}[c,d]$ be interval representations of $\mathbb{A}_n(\tau)$.*

1. *The dimension of* $\operatorname{Hom}(\mathbb{I}[a,b], \mathbb{I}[c,d])$ *as a K-vector space is either* 0 *or* 1.
2. *There exists a canonical basis $\{f^{c:d}_{a:b}\}$ for each nonzero* $\operatorname{Hom}(\mathbb{I}[a,b], \mathbb{I}[c,d])$ *such that*

$$(f^{c:d}_{a:b})_i = \begin{cases} 1_K : K \to K, & \text{if } i \in [a,b] \cap [c,d] \\ 0, & \text{otherwise.} \end{cases}$$

Proof. By the commutativity requirement on morphisms between representations, a nonzero morphism $g = \{g_i\} \in \operatorname{Hom}(\mathbb{I}[a,b], \mathbb{I}[c,d])$, if it exists, is completely determined by one of its internal morphisms, $g_j \in \operatorname{Hom}(K,K)$ for some fixed j in $[a,b] \cap [c,d]$. Since $\operatorname{Hom}(K,K)$ is of dimension 1, part 1 follows. The $f^{c:d}_{a:b}$ in part 2 is the g determined by $g_j = 1_K$. ◀

Each block $\Phi^{c:d}_{a:b} : \mathbb{I}[a,b]^{m_{a,b}} \to \mathbb{I}[c,d]^{m'_{c,d}}$ can be written in a matrix form where each entry is a morphism in $\operatorname{Hom}(\mathbb{I}[a,b], \mathbb{I}[c,d])$. Lemma 6 allows us to factor the common basis element of each entry and rewrite $\Phi^{c:d}_{a:b}$ using a K-matrix $M^{c:d}_{a:b}$ of size $m'_{c,d} \times m_{a,b}$:

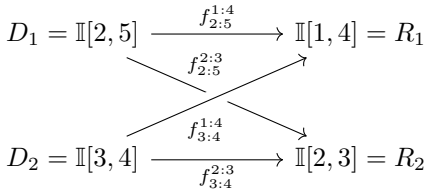
$$\Phi^{c:d}_{a:b} = \begin{cases} M^{c:d}_{a:b} f^{c:d}_{a:b}, & \text{if } \mathbb{I}[a,b] \trianglerighteq \mathbb{I}[c,d], \\ 0, & \text{otherwise} \end{cases}$$

3 Algebraic construction

Let us provide the construction for the commutative ladders of length 5. Details are provided for $\operatorname{rep} CL_5(fffff)$ using the formalism introduced for $\operatorname{arr}(\operatorname{rep} \vec{\mathbb{A}}_5)$. Slight adaptations to the construction make it work for any orientation τ.

3.1 Construction of a family of representations

Define the interval representations $D_1 = \mathbb{I}[2,5]$, $D_2 = \mathbb{I}[3,4]$, $R_1 = \mathbb{I}[1,4]$, and $R_2 = \mathbb{I}[2,3]$ of $\vec{\mathbb{A}}_5$. Note that for each of the four choices of ordered pairs (D_i, R_j), there exists a nonzero morphism $D_i \to R_j$, while no nonzero morphism exists between D_1 and D_2, and between R_1 and R_2. The directed graph with vertices given by the chosen intervals and arcs $x \to y$ defined by $x \trianglerighteq y$ is the complete bipartite directed graph $\vec{K}_{2,2}$.



Such a configuration is crucial to ensure that all four blocks can be nonzero in $\phi(d,\lambda)$ defined below, and to ensure indecomposability. Insight on how we find it is provided in the full version. The second ingredient we use is the Jordan cell $J_d(\lambda)$. Given $d \geq 1$ and $\lambda \in K$, $J_d(\lambda)$ is the matrix with value λ on the diagonal, 1 on the superdiagonal, and 0 elsewhere.

$$J_3(\lambda) = \begin{bmatrix} \lambda & 1 & 0 \\ 0 & \lambda & 1 \\ 0 & 0 & \lambda \end{bmatrix}$$

With this, we are ready to define an arrow $\phi(d,\lambda)$ and a representation $M(d,\lambda)$, which are identified to each other using Proposition 4 as $F(M(d,\lambda)) = \phi(d,\lambda)$.

▶ **Definition 7.** Let $d \geq 1$ and $\lambda \in K$.

1. We define the arrow $\phi(d,\lambda) \in \operatorname{arr}(\operatorname{rep} \vec{\mathbb{A}}_5)$ $\phi(d,\lambda) : \mathbb{I}[3,4]^d \oplus \mathbb{I}[2,5]^d \to \mathbb{I}[1,4]^d \oplus \mathbb{I}[2,3]^d$ by the matrix form $\phi(d,\lambda) = \begin{bmatrix} If_{3:4}^{1:4} & If_{2:5}^{1:4} \\ If_{3:4}^{2:3} & J_d(\lambda) f_{2:5}^{2:3} \end{bmatrix}$ where I is the $d \times d$ identity matrix.
2. We also define the representation $M(d,\lambda) \in \operatorname{rep} CL_5(ffff)$ by

$$M(d,\lambda): \quad \begin{array}{ccccccccc} K^d & \xrightarrow{\left[\begin{smallmatrix} I \\ 0 \end{smallmatrix}\right]} & K^{2d} & \xrightarrow{\mathrm{id}} & K^{2d} & \xrightarrow{[\,I\ 0\,]} & K^d & \longrightarrow & 0 \\ \uparrow & & \uparrow{\scriptstyle \left[\begin{smallmatrix} I \\ J_d(\lambda) \end{smallmatrix}\right]} & & \uparrow{\scriptstyle \left[\begin{smallmatrix} I & I \\ I & J_d(\lambda) \end{smallmatrix}\right]} & & \uparrow{\scriptstyle [\,I\ I\,]} & & \uparrow \\ 0 & \longrightarrow & K^d & \xrightarrow[\left[\begin{smallmatrix} 0 \\ I \end{smallmatrix}\right]]{} & K^{2d} & \xrightarrow[\mathrm{id}]{} & K^{2d} & \xrightarrow[[\,0\ I\,]]{} & K^d \end{array}$$

3.2 Indecomposability

We now show that the representations constructed above are indeed indecomposable and are pairwise non-isomorphic.

▶ **Theorem 8.** *Let $d \geq 1$ and $\lambda, \lambda' \in K$.*

1. *$M(d,\lambda)$ is indecomposable.*
2. *If $\lambda \neq \lambda'$ then $M(d,\lambda) \ncong M(d,\lambda')$.*

Proof. We check that $\operatorname{End} M(d,\lambda)$ is local. By Proposition 4, $\operatorname{End} M(d,\lambda) \cong \operatorname{End} \phi(d,\lambda)$. Letting (g_0, g_1) be an endomorphism of $\phi(d,\lambda)$, the diagram

$$\begin{array}{ccc} \mathbb{I}[3,4]^d \oplus \mathbb{I}[2,5]^d & \xrightarrow{\phi(d,\lambda)} & \mathbb{I}[1,4]^d \oplus \mathbb{I}[2,3]^d \\ \downarrow{\scriptstyle g_0} & & \downarrow{\scriptstyle g_1} \\ \mathbb{I}[3,4]^d \oplus \mathbb{I}[2,5]^d & \xrightarrow{\phi(d,\lambda)} & \mathbb{I}[1,4]^d \oplus \mathbb{I}[2,3]^d \end{array} \tag{1}$$

commutes. Then, g_0 and g_1 in matrix form with respect to the decompositions are

$$g_0 = \begin{bmatrix} Af_{3:4}^{3:4} & 0 \\ 0 & Bf_{2:5}^{2:5} \end{bmatrix} \text{ and } g_1 = \begin{bmatrix} Cf_{1:4}^{1:4} & 0 \\ 0 & Df_{2:3}^{2:3} \end{bmatrix}$$

where A, B, C, D are K-matrices of size $d \times d$. Since there are no nonzero morphisms from $\mathbb{I}[2,5]$ to $\mathbb{I}[3,4]$, nor vice versa, the off-diagonal entries of g_0 are 0. Likewise, there are no nonzero morphisms between $\mathbb{I}[1,4]$ and $\mathbb{I}[2,3]$ so the off-diagonal entries of g_1 are 0.

From the commutativity of Eq. (1), we get the equality

$$\begin{bmatrix} Af_{3:4}^{1:4} & Af_{2:5}^{1:4} \\ Bf_{3:4}^{2:3} & BJ_d(\lambda)f_{2:5}^{2:3} \end{bmatrix} = \begin{bmatrix} Cf_{3:4}^{1:4} & Df_{2:5}^{1:4} \\ Cf_{3:4}^{2:3} & J_d(\lambda)Df_{2:5}^{2:3} \end{bmatrix}$$

which implies that $A = B = C = D$ and $AJ_d(\lambda) = J_d(\lambda)A$ as K-matrices since the morphisms $f_{a:b}^{c:d}$ appearing above are nonzero. We infer that

$$\operatorname{End}\phi(d,\lambda) \cong \{A \in K^{d\times d} \mid AJ_d(\lambda) = J_d(\lambda)A\}.$$

A direct computation shows that A is a member of the above ring if and only if A is upper triangular Toeplitz:

$$A = \begin{bmatrix} a_1 & a_2 & \dots & a_{d-1} & a_d \\ 0 & a_1 & a_2 & \dots & a_{d-1} \\ & & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & a_1 & a_2 \\ 0 & \dots & 0 & 0 & a_1 \end{bmatrix}.$$

The matrix A is invertible if and only if a_1 is nonzero, and so for any A, either A or $I - A$ is invertible. Thus, $\operatorname{End}\phi(d,\lambda)$ is local and $M(d,\lambda)$ is indecomposable.

By a similar computation, $M(d,\lambda) \cong M(d,\lambda')$ implies that there is some invertible matrix A such that $AJ_d(\lambda) = J_d(\lambda')A$ which is impossible when $\lambda \neq \lambda'$. ◀

Proposition 8 together with the observation that if $d \neq d'$ then $M(d,\lambda) \ncong M(d',\lambda')$ provides an easy proof for the following corollary when $\tau = ffff$. Moreover, the method above can be used to produce similar examples for any orientation τ on length $n = 5$ by finding a similar configuration isomorphic to $\vec{K}_{2,2}$ from the intervals and arcs determined by $\trianglerighteq$. As a result, we get a constructive proof of the following result as a corollary.

▶ **Corollary 9.** *For any $n \geq 5$ and orientation τ, the commutative ladder $CL_n(\tau)$ is representation infinite.*

4 Realizations

We give realizations of our indecomposable persistence modules for $\lambda = 0$, first relying purely on topological spaces and then using a geometric Vietoris-Rips construction.

4.1 Topological construction

Given $d \geq 1$, we build a diagram $\mathbb{S}(d)$ of topological spaces and inclusions. The spaces in the middle column take the form of a sandal consisting of a planar sole and a set of d straps. Other spaces are either missing some edges or have some faces filled in. Figure 1 presents the complete realization.

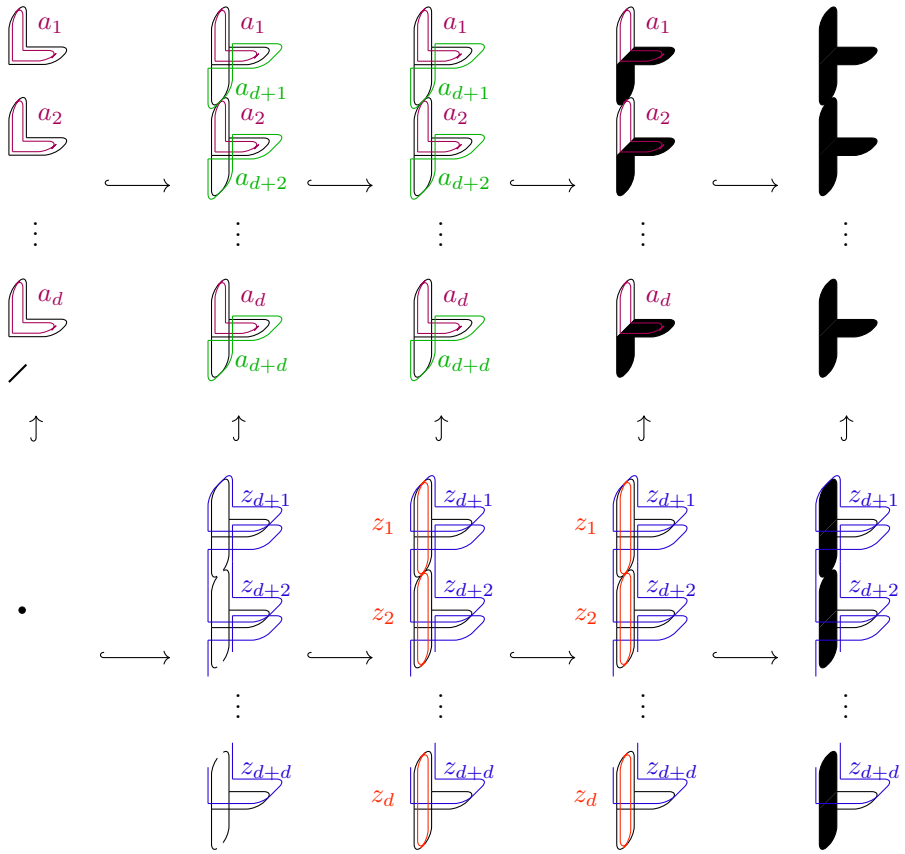


Figure 1 Diagram of spaces; and representatives for homology bases (in color).

The resulting diagram of spaces has maps that are all inclusions, and therefore all squares commute. Using the singular homology functor with coefficient in field K, we obtain a representation $H_1(\mathbb{S}(d))$ of $CL_5(ffff)$.

▶ **Theorem 10.**

$H_1(\mathbb{S}(d)) \cong M(d, 0)$

Proof. Relative to the choice of bases indicated in Fig. 1, the induced maps have the same matrix forms as the matrices in $M(d, 0)$. ◀

4.2 Vietoris-Rips construction

Next, we use the well-known Vietoris-Rips construction to build simplicial complexes having the suitable topology. Recall that the Vietoris-Rips complex $V(P, r)$ is the clique complex of the set of all edges that can be formed from points $p \in P$ with length less than $2r$. Note that, for $r \leq r'$, $V(P, r) \subset V(P, r')$, and hence we have a filtration.

In our construction, we provide two different points sets P_ℓ and P_u corresponding to the lower and upper rows. The point sets P_ℓ and P_u are built by assembling what we call *tiles* in a regular pattern. The union (possibly sharing common points along the edges) of translations and reflections of these tiles defines the global point sets.

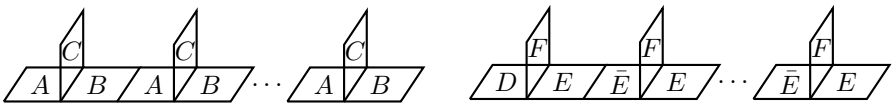


Figure 2 Assembling of the tiles with the upper row on the left.

We define three types of tiles, labelled A, B, C for the upper row and three, D, E, F for the lower row. The tile denoted $\bar{E}$ is obtained by reflection of tile E and we call it the reversed E tile. These tiles are arranged as in Fig. 2 to obtain P_u and P_ℓ, respectively. Note that the lower row presents an asymmetry as the D tile is only used once. Details on the content of the tiles and additional figures can be found in the full version.

We then choose five radius parameters $r_1, \ldots, r_5$ so that the diagram

$$\begin{array}{ccccccccc} |V(P_u,r_1)| & \longrightarrow & |V(P_u,r_2)| & \longrightarrow & |V(P_u,r_3)| & \longrightarrow & |V(P_u,r_4)| & \longrightarrow & |V(P_u,r_5)| \\ \uparrow{\scriptstyle f_1} & & \uparrow{\scriptstyle f_2} & & \uparrow{\scriptstyle f_3} & & \uparrow{\scriptstyle f_4} & & \uparrow{\scriptstyle f_5} \\ |V(P_\ell,r_1)| & \longrightarrow & |V(P_\ell,r_2)| & \longrightarrow & |V(P_\ell,r_3)| & \longrightarrow & |V(P_\ell,r_4)| & \longrightarrow & |V(P_\ell,r_5)| \end{array}$$

of topological spaces and continuous maps has the same homology as $\mathbb{S}(d)$, where $|V(P,r)|$ denotes underlying topological space. The vertical maps are "obvious" ones that give the same induced maps as the inclusions in Figure 1. Details for the spaces and maps are provided in the full version. The resulting diagram is displayed in Figure 3. For clarity of the illustration, we omit some extra edges and triangles not affecting homology.

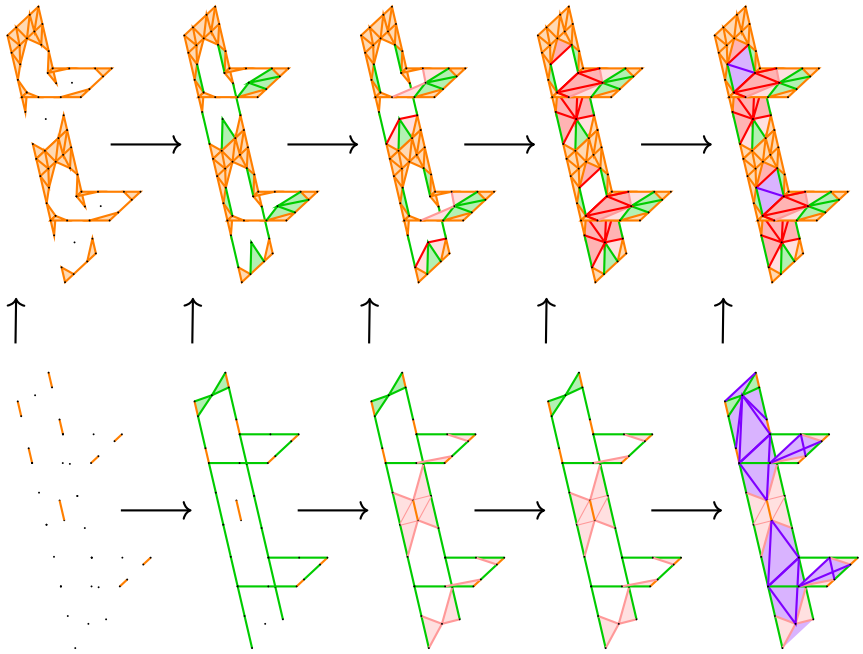


Figure 3 Complete realization, $d = 2$ case

Importantly, our construction does not rely on degeneracy. With P_ℓ and P_u fixed, we can freely choose the radius parameters r_i within intervals (x_i, y_i) of non-zero width. Dually, small perturbations of the point sets do not change the homology of the construction. Being even more restrictive, we can find similar intervals $I_i \subset (x_i, y_i)$ with positive width where there are no changes in the filtration. We get the following.

▶ **Lemma 11.** *Let ρ be the minimum of the diameters of I_i, and fix each r_i to be the center of I_i, for $i = 1, \dots, 5$. Replacing each point of our input by a point within a ball of radius $\frac{\rho}{2}$ around it does not change the topology.*

Proof. By construction, for any i, there are no edges of length l such that $2r_i - \rho < l < 2r_i + \rho$.

We now replace every point of P by a point located within distance $\frac{\rho}{2}$. Note that the pairwise distance after this addition of noise are modified by at most ρ. Therefore, the complexes for radii r_i are unaffected as no pairwise distance can cross that threshold. ◀

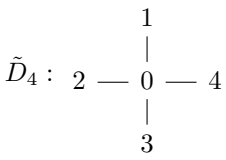
5 Other elementary grids

Previously we considered the 2×5 grid. We now consider the two other elementary representation infinite grids, the 3×3 grid and the $2 \times 2 \times 2$ cube. With this, we will have constructions for all possible representation infinite commutative grids, via embedding.

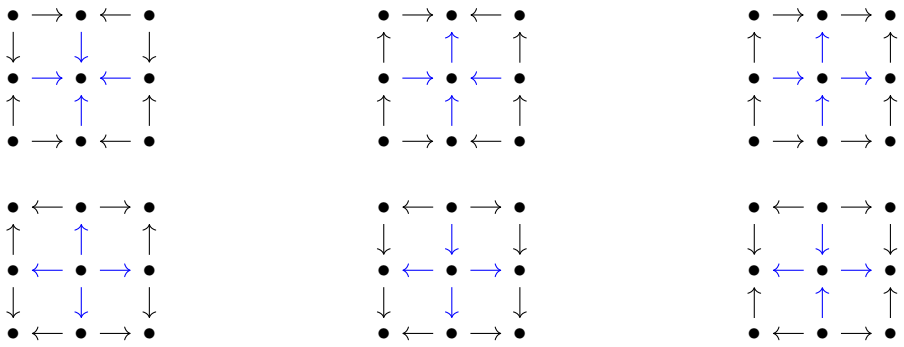
5.1 Three by three grid

The Euclidean quiver of type $\tilde{D}_4$ is representation infinite for any orientation of the arrows.

$\tilde{D}_4$:



We build infinite family of indecomposables for the $\tilde{D}_4$-type quiver with appropriate orientation and complete it to be indecomposable representations of a 3×3 grid. Up to symmetries, there are six different orientations of the 3×3 grid. We classify them according to the resulting orientation of the central $\tilde{D}_4$, shown in Fig. 4.

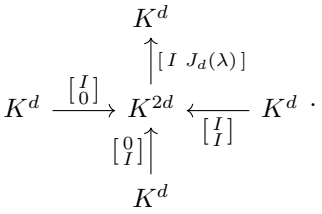


(a) $\tilde{D}_4$ same direction **(b)** 3 in, 1 out; and dual **(c)** 2 in, 2 out

Figure 4 Configurations for the 3×3 grid.

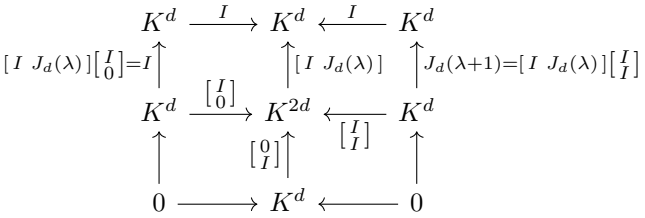
We fix the vector spaces to be K^{2d} at the central vertex and K^d elsewhere. Note that at least two arrows of $\tilde{D}_4$ will be pointing in the same direction relative to the central vertex. On two of these arrows, we assign matrices $\begin{bmatrix} I & 0 \end{bmatrix}$ and $\begin{bmatrix} 0 & I \end{bmatrix}$ or $\begin{bmatrix} I \\ 0 \end{bmatrix}$ and $\begin{bmatrix} 0 \\ I \end{bmatrix}$ depending on their orientation. The remaining two arrows are assigned $\begin{bmatrix} I & J_d(\lambda) \end{bmatrix}$ or its transpose, and $\begin{bmatrix} I & I \end{bmatrix}$ or

its transpose. For example, with three arrows pointing in and one out, we can have:



The proof of indecomposability is again by computation of the endomorphism ring. Let $f = (f_0, \ldots, f_4)$ be an endomorphism. In matrix form, $f_0 = \left[\begin{smallmatrix} A & B \\ C & D \end{smallmatrix}\right] : K^{2d} \to K^{2d}$, where 0 is the central vertex. Without loss of generality, we assume that the pair of arrows pointing in the same direction and assigned matrices $\left[\begin{smallmatrix} I \\ 0 \end{smallmatrix}\right]$ and $\left[\begin{smallmatrix} 0 \\ I \end{smallmatrix}\right]$ (or $\left[\begin{smallmatrix} I & 0 \end{smallmatrix}\right]$ and $\left[\begin{smallmatrix} 0 & I \end{smallmatrix}\right]$), start from (or point towards) vertices 1 and 2, respectively. From commutativity requirement for endomorphisms, $f_0 = \left[\begin{smallmatrix} A & B \\ C & D \end{smallmatrix}\right] = \left[\begin{smallmatrix} f_1 & 0 \\ 0 & f_2 \end{smallmatrix}\right]$. Suppose that the arrow assigned $\left[\begin{smallmatrix} I & I \end{smallmatrix}\right]$ (or its transpose) points to (or starts from) vertex 3. The commutativity requirement with f_3 then requires $f_1 = f_3 = f_2$. Final commutativity requirement then forces $f_1 = f_2 = f_3 = f_4$ and $f_1 J_d(\lambda) = J_d(\lambda) f_1$. Thus, the endomorphism ring is local.

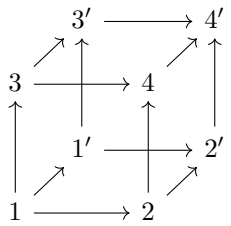
Completing the example into the 3×3 grid is easy. In the squares where the representation $\tilde{D}_4$ provides a nonzero composition of maps, we use that composition on one arrow and the identity on the other arrow. Otherwise we simply use a 0 vector space and 0 maps.



This algebraic construction can then be realized through a diagram of topological spaces and inclusions using our previous *sandal* construction. Details are provided in the full version.

5.2 Commutative cube

The commutative cube C is defined to be the quiver



bound by commutativity relations. Similar to Lemma 4, it can be checked that $\operatorname{rep} C \cong \operatorname{arr}(\operatorname{rep} CL_2(f))$. Thus, we write a representation of C as a morphism between representations of $CL_2(f)$ by taking the morphism from the front face to the back face.

In $\operatorname{rep}(CL_2(f))$, an analogue of Lemma 6 does not hold. In particular, the indecomposables

I_1, I_2 in $\mathrm{rep}(CL_2(f))$ given by:

$$I_1: \begin{array}{ccc} K & \xrightarrow{1} & K \\ \uparrow & & \uparrow{\scriptstyle 1} \\ 0 & \longrightarrow & K \end{array} \quad \text{and} \quad I_2: \begin{array}{ccc} K & \longrightarrow & 0 \\ {\scriptstyle 1}\uparrow & & \uparrow \\ K & \xrightarrow{1} & K \end{array}$$

have $\dim \mathrm{Hom}(I_1, I_2) = 2$. The vector space $\mathrm{Hom}(I_1, I_2)$ can be given the basis $\{f_2, f_3\}$, where f_2 is the identity on the lower right corner and zero elsewhere, and f_3 is the identity on the upper left corner and zero elsewhere.

Intuitively, we see that this is related to representations of the *Kronecker quiver* $Q_2 : 1 \rightrightarrows 2$ by thinking about the two arrows as the linearly independent f_2, f_3. This statement can be made precise by the following.

▶ **Theorem 12.** *There is a fully faithful K-functor $\theta : \mathrm{rep}\, Q_2 \to \mathrm{rep}\, C$ that preserves indecomposability and isomorphism classes, where θ takes a representation $V : V_1 \overset{g_1}{\underset{g_2}{\rightrightarrows}} V_2$ to*

$$\theta(V): \begin{array}{ccccc} & & V_2 & \longrightarrow & 0 \\ & {\scriptstyle g_2}\nearrow & \uparrow & \nearrow & \uparrow \\ V_1 & \xrightarrow{1} & V_1 & & \\ \uparrow{\scriptstyle 1} & & \uparrow & & \\ V_2 & \xrightarrow{1} & V_2 & & \\ \uparrow & \nearrow{\scriptstyle 1} & \uparrow{\scriptstyle g_1} & & \\ 0 & \longrightarrow & V_1 & & \end{array}$$

and a morphism $\phi = (\phi_1, \phi_2) : V \to W$ to $(0, \phi_1, \phi_1, \phi_1, \phi_2, \phi_2, \phi_2, 0) : \theta(V) \to \theta(W)$, where these maps are specified for the vertices in the order $1, \ldots, 4, 1', \ldots, 4'$.

Proof. That θ is a K-functor is easy to check. By the definition, $\theta(\phi) = 0$ implies that $\phi = 0$. Thus, θ is faithful. To see that θ is full, let V be as above, $W : W_1 \overset{g_1'}{\underset{g_2'}{\rightrightarrows}} W_2$ and

$$\alpha = (\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_{1'}, \alpha_{2'}, \alpha_{3'}, \alpha_{4'}) : \theta(V) \to \theta(W)$$

be a morphism. Then, $\alpha_1 : 0 \to 0$ and $\alpha_{4'} : 0 \to 0$ are zero maps by the forms of $\theta(V)$ and $\theta(W)$. The commutativity requirements for morphisms imply that $\alpha_2 = \alpha_3 = \alpha_4$ and $\alpha_{2'} = \alpha_{3'} = \alpha_{4'}$, and furthermore, $\alpha_{3'} g_2 = g_2' \alpha_3$ and $\alpha_{2'} g_1 = g_1' \alpha_2$. Thus, $(\alpha_2, \alpha_{2'})$ is a morphism from V to W such that $\theta((\alpha_2, \alpha_{2'})) = \alpha$.

Let $V \in \mathrm{rep}\, Q_2$ be indecomposable. By part 2 of Lemma 2, $\mathrm{End}\, V$ is local. By the above result that θ is fully faithful, $\mathrm{End}\, \theta(V) \cong \mathrm{End}\, V$, and so $\mathrm{End}\, \theta(V)$ is local. Therefore, $\theta(V) \in \mathrm{rep}\, C$ is indecomposable by part 1 of Lemma 2.

If $\alpha : \theta(V) \to \theta(W)$ is an isomorphism with inverse β then $(\alpha_2, \alpha_{2'}) : V \to W$ is an isomorphism with inverse $(\beta_2, \beta_{2'})$. Thus, if $\theta(V) \cong \theta(W)$, then $V \cong W$. This shows that θ preserves isomorphism classes. ◀

The quiver Q_2 is representation infinite, and its indecomposable representations are well-known. See for example Proposition 1.6 of [3]. In particular, one example of an infinite family is the regular indecomposables $R_n(\lambda) : K^n \overset{I}{\underset{J_n(\lambda)}{\rightrightarrows}} K^n$ for $n \geq 0$ and $\lambda \in K$. The following corollary is immediate from Theorem 12.

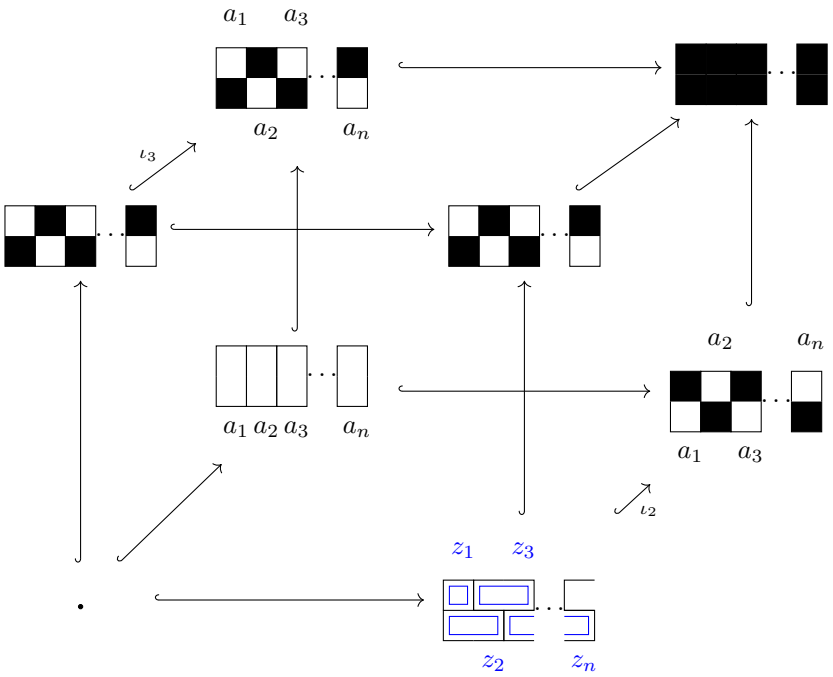
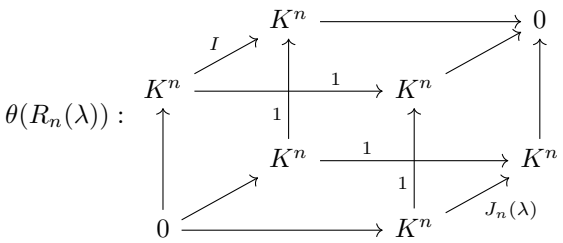


Figure 5 Topological realization of $\theta(R_n(0))$.

▶ **Corollary 13.** *The commutative cube C is representation infinite.*

By the above arguments, $\theta(R_n(\lambda))$:



are indecomposable and pairwise non-isomorphic for $n \geq 0$. We give a topological realization for $\theta(R_n(0))$ in Fig. 5. In the back face, we have n half filled-in strips arranged side by side horizontally in an alternating pattern (Fig. 5 shows the case n even). Using the lower left corner, we are able to flip the pattern. Coming from the front face, and with the given choice of basis, the induced map $H_1(\iota_2)$ is $J_n(0)$ while $H_1(\iota_3)$ is the identity I.

6 Discussion

We have illustrated constructions of infinite families of indecomposable persistence modules together with topological realizations over the small commutative grids. By embedding, this provides constructions for all possible representation infinite commutative grids.

In addition to our families of indecomposables, other parametrized families might be of interest. More generally, for representation tame commutative grids, could we realize parametrized families that generate all indecomposables?

References

1 Hideto Asashiba, Emerson G. Escolar, Yasuaki Hiraoka, and Hiroshi Takeuchi. Matrix method for persistence modules on commutative ladders of finite type. *arXiv preprint arXiv:1706.10027*, 2017.

2 Ibrahim Assem, Andrzej Skowronski, and Daniel Simson. *Elements of the Representation Theory of Associative Algebras: Volume 1: Techniques of Representation Theory*, volume 65. Cambridge University Press, 2006.

3 Michael Barot. *Introduction to the representation theory of algebras*. Springer, 2014.

4 Gunnar Carlsson and Vin de Silva. Zigzag persistence. *Foundations of computational mathematics*, 10(4):367–405, 2010.

5 Gunnar Carlsson and Afra Zomorodian. The theory of multidimensional persistence. *Discrete & Computational Geometry*, 42(1):71–93, 2009.

6 Lorin Crawford, Anthea Monod, Andrew X. Chen, Sayan Mukherjee, and Raúl Rabadán. Topological summaries of tumor images improve prediction of disease free survival in glioblastoma multiforme. *arXiv preprint arXiv:1611.06818*, 2016.

7 Mary-Lee Dequeant, Sebastian Ahnert, Herbert Edelsbrunner, Thomas M. A. Fink, Earl F. Glynn, Gaye Hattem, Andrzej Kudlicki, Yuriy Mileyko, Jason Morton, Arcady R. Mushegian, et al. Comparison of pattern detection methods in microarray time series of the segmentation clock. *PLoS One*, 3(8):e2856, 2008.

8 Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. *Discrete Comput Geom*, 28:511–533, 2002.

9 Emerson G. Escolar and Yasuaki Hiraoka. Persistence modules on commutative ladders of finite type. *Discrete & Computational Geometry*, 55(1):100–157, 2016.

10 Chad Giusti, Eva Pastalkova, Carina Curto, and Vladimir Itskov. Clique topology reveals intrinsic geometric structure in neural correlations. *Proceedings of the National Academy of Sciences*, 112(44):13455–13460, 2015.

11 Heather A Harrington, Nina Otter, Hal Schenck, and Ulrike Tillmann. Stratifying multiparameter persistent homology. *arXiv preprint arXiv:1708.07390*, 2017.

12 Yasuaki Hiraoka, Takenobu Nakamura, Akihiko Hirata, Emerson G. Escolar, Kaname Matsue, and Yasumasa Nishiura. Hierarchical structures of amorphous solids characterized by persistent homology. *Proceedings of the National Academy of Sciences*, 113(26):7035–7040, 2016.

13 Lida Kanari, Paweł Dłotko, Martina Scolamiero, Ran Levi, Julian Shillcock, Kathryn Hess, and Henry Markram. A topological representation of branching neuronal morphologies. *Neuroinformatics*, pages 1–11, 2017.

14 Yongjin Lee, Senja D. Barthel, Paweł Dłotko, S. Mohamad Moosavi, Kathryn Hess, and Berend Smit. Quantifying similarity of pore-geometry in nanoporous materials. *Nature Communications*, 8, 2017.

15 Michael Lesnick and Matthew Wright. Interactive visualization of 2-d persistence modules. *arXiv preprint arXiv:1512.00180*, 2015.

16 James R. Munkres. *Elements of algebraic topology*, volume 2. Addison-Wesley Menlo Park, 1984.

17 Monica Nicolau, Arnold J. Levine, and Gunnar Carlsson. Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival. *Proceedings of the National Academy of Sciences*, 108(17):7265–7270, 2011.

Approximating the Distribution of the Median and other Robust Estimators on Uncertain Data

Kevin Buchin
Department of Mathematics and Computer Science, TU Eindhoven
Eindhoven, The Netherlands
k.a.buchin@tue.nl
https://orcid.org/0000-0002-3022-7877

Jeff M. Phillips
School of Computing, University of Utah
Salt Lake City, USA
jeffp@cs.utah.edu

Pingfan Tang
School of Computing, University of Utah
Salt Lake City, USA
tang1984@cs.utah.edu

Abstract

Robust estimators, like the median of a point set, are important for data analysis in the presence of outliers. We study robust estimators for locationally uncertain points with discrete distributions. That is, each point in a data set has a discrete probability distribution describing its location. The probabilistic nature of uncertain data makes it challenging to compute such estimators, since the true value of the estimator is now described by a distribution rather than a single point. We show how to construct and estimate the distribution of the median of a point set. Building the approximate support of the distribution takes near-linear time, and assigning probability to that support takes quadratic time. We also develop a general approximation technique for distributions of robust estimators with respect to ranges with bounded VC dimension. This includes the geometric median for high dimensions and the Siegel estimator for linear regression.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Uncertain Data, Robust Estimators, Geometric Median, Tukey Median

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.16

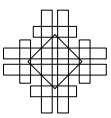
Related Version A full version of this paper is available at `https://arxiv.org/abs/1601.00630`

Funding NSF CCF-1115677, CCF-1350888, IIS-1251019, ACI-1443046, CNS-1514520, CNS-1564287, and NWO project no. 612.001.207

1 Introduction

Most statistical or machine learning models of noisy data start with the assumption that a data set is drawn iid (independent and identically distributed) from a single distribution. Such distributions often represent some true phenomenon under some noisy observation. Therefore, approaches that mitigate the influence of noise, involving robust statistics or regularization, have become commonplace.

However, many modern data sets are clearly not generated iid, rather each data element represents a separate object or a region of a more complex phenomenon. For instance, each

34th International Symposium on Computational Geometry (SoCG 2018).
Editors: Bettina Speckmann and Csaba D. Tóth; Article No. 16; pp. 16:1–16:14


Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

data element may represent a distinct person in a population or an hourly temperature reading. Yet, this data can still be noisy; for instance, multiple GPS locational estimates of a person, or multiple temperature sensors in a city. The set of data elements may be noisy *and* there may be multiple inconsistent readings of each element. To model this noise, the inconsistent readings can naturally be interpreted as a probability distribution.

Given such locationally noisy, non-iid data sets, there are many unresolved and important analysis tasks ranging from classification to regression to summarization. In this paper, we initiate the study of robust estimators [11, 18] on locationally uncertain data. More precisely, we consider an input data set of size n, where each data point's location is described by a discrete probability distribution. We will assume these discrete distributions have a support of at most k points in $\mathbb{R}^d$; and for concreteness and simplicity we will focus on cases where each point has support described by exactly k points, each being equally likely.

Although algorithms for locationally uncertain points have been studied in quite a few contexts over the last decade [10, 16, 13, 4, 12, 3, 1, 2, 23] (see more through discussion in full version [8]), few have directly addressed the problem of noise in the data. As the uncertainty is often the direct consequence of noise in the data collection process, this is a pressing concern. As such we initiate this study focusing on the most basic robust estimators: the median for data in $\mathbb{R}^1$, as well as its generalization the geometric median and the Tukey median for data in $\mathbb{R}^d$, defined in Section 1.1. Being robust refers to the fact that the median and geometric medians have a *breakdown point*s of 0.5, that is, if less than 50% of the data points (the outliers) are moved from the true distribution to some location infinitely far away, the estimator remains within the extent of the true distribution [17]. The Tukey median has a breakdown point between $\frac{1}{d+1}$ and $\frac{1}{3}$ [5].

In this paper, we generalize the median (and other robust estimators) to locationally uncertain data, where the outliers can occur not just among the n data points, but also as part of the discrete distributions representing their possible locations.

The main challenge is in modeling these robust estimators. As we do not have precise locations of the data, there is not a single minimizer of $\mathsf{cost}(x, Q)$; rather there may be as many as k^n possible input point sets Q (the combination of all possible locations of the data). And the expected value of such a minimizer is not robust in the same way that the mean is not robust. As such we build a distribution over the possible locations of these cost-minimizers. In $\mathbb{R}^1$ (by defining boundary cases carefully) this distribution is of size at most $O(nk)$, the size of the input, but already in $\mathbb{R}^2$ it may be as large as k^n.

Our results. We design algorithms to create an approximate support of these median distributions. We create small sets T (called an *ε-support*) such that each possible median m_Q from a possible point set Q is within a distance $\varepsilon \cdot \mathsf{cost}(m_Q, Q)$ of some $x \in T$. In $\mathbb{R}$ we can create a support set T of size $O(k/\varepsilon)$ in $O(nk \log(nk))$ time. We show that the bound $O(k/\varepsilon)$ is tight since there may be k large enough modes of these distributions, each requiring $\Omega(1/\varepsilon)$ points to represent. In $\mathbb{R}^d$ our bound on $|T|$ is $O(k^d/\varepsilon^d)$, for the Tukey median and the geometric median. If we do not need to cover sets of medians m_Q which occur with probability less than ε, we can get a bound $O(d/\varepsilon^2)$ in $\mathbb{R}^d$. In fact, this general approach in $\mathbb{R}^d$ extends to other estimators, including the Siegel estimator [19] for linear regression. We then need to map weights onto this support set T. We can do so exactly in $O(n^2 k)$ time in $\mathbb{R}^1$ or approximately in $O(1/\varepsilon^2)$ time in $\mathbb{R}^d$.

Another goal may be to then construct a single-point estimator of these distributions: the median of these median distributions. In $\mathbb{R}^1$ we can show that this process is stable up to $\mathsf{cost}(m_Q, Q)$ where m_Q is the resulting single-point estimate. However, we also show

that already in $\mathbb{R}^1$ such estimators are not stable with respect to the weights in the median distribution, and hence not stable with respect to the probability of any possible location of an uncertain point. That is, infinitesimal changes to such probabilities can greatly change the location of the single-point estimator. As such, we argue the approximate median distribution (which is stable with respect to these changes) is the best robust representation of such data.

1.1 Formalization of model and notation

We consider a set of n locationally uncertain points $\mathcal{P} = \{P_1, \ldots, P_n\}$ so that each P_i has k possible locations $\{p_{i,1}, \ldots, p_{i,k}\} \subset \mathbb{R}^d$. Here, $P_i = \{p_{i,1}, \ldots, p_{i,k}\}$ is a multiset, which means a point in P_i may appear more than once. Let $P_{\mathsf{flat}} = \cup_i \{p_{i,1}, \ldots, p_{i,k}\}$ represent all positions of all points in $\mathcal{P}$, which implies P_{flat} is also a multiset. We consider each $p_{i,j}$ to be an equally likely (with probability $1/k$) location of P_i, and can extend our techniques to non-uniform probabilities and uncertain points with fewer than k possible locations. For an uncertain point set $\mathcal{P}$ we say $Q \Subset \mathcal{P}$ is a *traversal* of $\mathcal{P}$ if $Q = \{q_1, \ldots q_n\}$ has each q_i in the domain of P_i (e.g., $q_i = p_{i,j}$ for some j). We denote by $\Pr_{Q \Subset \mathcal{P}}[\gamma(Q)]$ the probability of the event $\gamma(Q)$, given that Q is a randomly selected traversal from $\mathcal{P}$, where the selection of each q_i from P_i is independent of $q_{i'}$ from $P_{i'}$.

We are particularly interested in the case where n is large and k is small. For technical simplicity we assume an extended RAM model where k^n (the number of possible traversals of point sets) can be computed in $O(1)$ time and fits in $O(1)$ words of space.

We consider three definitions of medians. In one dimension, given a set $Q = \{q_1, q_2, \ldots, q_n\}$ that w.l.o.g. satisfies $q_1 \leq q_2 \leq \ldots \leq q_n$, we define the *median* m_Q as $q_{\frac{n+1}{2}}$ when n is odd and $q_{\frac{n}{2}}$ when n is even. There are several ways to generalize the median to higher dimensions [5], herein we focus on the geometric median and Tukey median. Define $\mathsf{cost}(x, Q) = \frac{1}{n}\sum_{i=1}^{n} \|x - q_i\|$ where $\|\cdot\|$ is the Euclidian norm. Given a set $Q = \{q_1, q_2, \ldots, q_n\} \subset \mathbb{R}^d$, the *geometric median* is defined as $m_Q = \arg\min_{x \in \mathbb{R}^d} \mathsf{cost}(x, Q)$. The Tukey depth [20] of a point p with respect to a set $Q \subset \mathbb{R}^d$ is defined $\mathsf{depth}_Q(p) := \min_{H \in \mathcal{H}_p} |H \cap Q|$ where $\mathcal{H}_p := \{H \text{ is a closed halfspace in } \mathbb{R}^d \mid p \in H\}$. Then a *Tukey median* of a set Q is a point p that can maximize the Tukey depth.

2 Constructing a single point estimate

We begin by exploring the construction of a single point estimator of set of n locationally uncertain points $\mathcal{P}$. We demonstrate that while the estimator is stable with respect to the value of cost, the actual minimum of that function is not stable and provides an incomplete picture for multimodal uncertainties.

It is easiest to explore this through a weighted point set $X \subset \mathbb{R}^1$. Given a probability distribution defined by $\omega : X \to [0, 1]$, we can compute its weighted median by scanning from smallest to largest until the sum of weights reaches 0.5.

There are two situations whereby we obtain such a discrete weighted domain. The first domain is the set T of possible locations of medians under different instantiations of the uncertain points with weights $\hat{w}$ as the probability of those medians being realized; see constructions in Section 3.2 and Section 3.5. Let the resulting weighted median of $(T, \hat{w})$ be m_T. The second domain is simply the set P_{flat} of all possible locations of $\mathcal{P}$, and its weight w where $w(p_{i,j})$ is the fraction of $Q \Subset \mathcal{P}$ which take $p_{i,j}$ as their median (possibly 0). Let the weighted median of (P_{flat}, w) be $m_{\mathcal{P}}$.

▶ **Theorem 1.** *$|m_T - m_{\mathcal{P}}| \leq \varepsilon \mathsf{cost}(m_{\mathcal{P}}) \leq \varepsilon \mathsf{cost}(m_Q, Q)$, $Q \Subset \mathcal{P}$ is any traversal with $m_{\mathcal{P}}$ as its median.*

Proof. We can divide $\mathbb{R}$ into $|T|$ intervals, one associated with each $x \in T$, as follows. Each $z \in \mathbb{R}$ is in an interval associated with $x \in T$ if z is closer to x than any other point $y \in T$, unless $|z - y| \leq \varepsilon \mathsf{cost}(z)$ but $|z - x| > \mathsf{cost}(z)$. Thus a point $p_{i,j}$ whose weight $w(p_{i,j})$ contributes to $\hat{w}(x)$, is in the interval associated with x.

Thus, if $p_{i,j} = m_{\mathcal{P}}$, then the sum of all weights of all points greater than $p_{i,j}$ is at most 0.5, and the sum of all weights of points less than $p_{i,j}$ is less than 0.5. Hence if $m_{\mathcal{P}}$ is in an interval associated with $x \in T$, then the sum of all weights of points $p_{i,j}$ in intervals greater than that of x must be at most 0.5 and those less than that of x must be less than 0.5. Hence $m_T = x$, and $|x - p_{i,j}| \leq \varepsilon \mathsf{cost}(m_{\mathcal{P}})$ as desired. ◀

Non-robustness of single point estimates. The geometric median of the set $\{m_Q$ is a geometric median of $Q \mid Q \Subset \mathcal{P}\}$ is not stable under small perturbations in weights; it stays within the convex hull of the set, but otherwise not much can be said, even in $\mathbb{R}^1$. Consider the example with $n = 3$ and $k = 2$, where $p_{1,1} = p_{1,2} = p_{2,1} = 0$ and $p_{2,2} = p_{3,1} = p_{3,2} = \Delta$ for some arbitrary Δ. The median will be at 0 or Δ, each with probability $1/2$, depending on the location of P_2. We can also create a more intricate example where $\hat{\mathsf{cost}}(0) = \hat{\mathsf{cost}}(\Delta) = 0$. As these examples have m_Q at 0 or Δ equally likely with probability $1/2$, then canonically in $\mathbb{R}^1$ we would have the median of this distribution at 0, but a slight change in probability (say from sampling) could put it all the way at Δ. This indicates that a representation of the distribution of medians as we study in the remainder is more appropriate for noisy data.

3 Approximating the median distribution

The big challenge in constructing an ε-support T is finding the points $x \in P_{\mathsf{flat}}$ which have small values of $\mathsf{cost}(x, Q)$ (recall $\mathsf{cost}(x, Q) = \frac{1}{n}\sum_{i=1}^{n} \|x - q_i\|$) for some $Q \Subset \mathcal{P}$. But this requires determining the smallest cost $Q \Subset \mathcal{P}$ that has $x \in Q$ and x is the median of Q.

One may think (as the authors initially did) that one could simply use a proxy function $\hat{\mathsf{cost}}(x) = \frac{1}{n}\sum_{i=1}^{n} \min_{1 \leq j \leq k} \|x - p_{i,j}\|$, which is relatively simple to compute as the lower envelope of cost functions for each P_i. Clearly $\hat{\mathsf{cost}}(x) \leq \mathsf{cost}(x, Q)$ for all $Q \Subset \mathcal{P}$, so a set $\hat{T}$ satisfying a similar approximation for $\hat{\mathsf{cost}}$ will satisfy our goals for cost. However, there exist (rather adversarial) data sets $\mathcal{P}$ where $\hat{T}$ would require $\Omega(nk)$ points; see full version [8]. On the other hand, we show this is not true for cost. The key difference between cost and $\hat{\mathsf{cost}}$ is that $\hat{\mathsf{cost}}$ does not enforce the use of some $Q \Subset \mathcal{P}$ of which x is a median. That is, that (roughly) half the points are to the left and half to the right for this Q.

Proxy functions L, R, and D. We handle this problem by first introducing two families of functions, defined precisely shortly. We let $L_i(x)$ (resp. $R_i(x)$) represent the contribution to cost at x from the closest possible location $p_{i,j}$ of an uncertain point P_i to the left (resp. right) of x. This allows us to decompose the elements of this cost. However, it does not help us to enforce this balance. Hence we introduce a third proxy function

$$D_i(x) = L_i(x) - R_i(x)$$

capturing the difference between L_i and R_i. We will show that the choice of which points are used on the left or right of x is completely determined by the D_i values. In particular, we maintain the D_i values (for all $i \in [n]$) in sorted order, and use the i with larger D_i values on the right, and smaller D_i values on the left for the min cost $Q \Subset \mathcal{P}$.

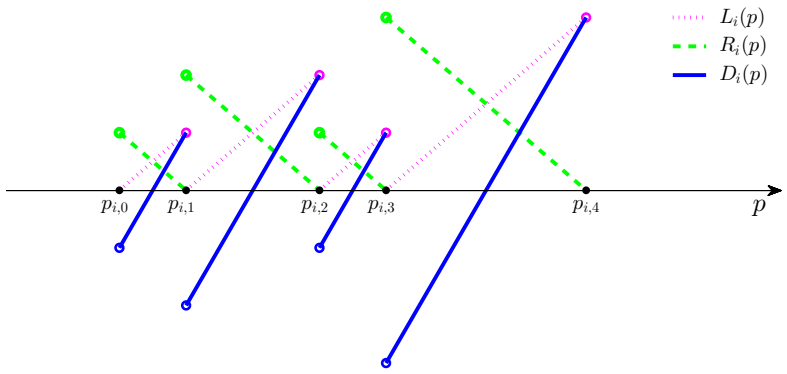


Figure 1 The plot of $L_i(p)$, $R_i(p)$ and $D_i(p)$.

To define L_i, R_i, and D_i, we first assume that P_{flat} and P_i for all $i \in [n]$ are sorted (this would take $O(nk \log(nk))$ time). Then to simplify definitions we add two dummy points to each P_i, and introduce the notation $\widetilde{P}_i = P_i \cup \{p_{i,0}, p_{i,k+1}\}$ and $\widetilde{\mathcal{P}} = \{\widetilde{P}_1, \widetilde{P}_2, \cdots, \widetilde{P}_n\}$, where $p_{i,0} = \min P_{\mathsf{flat}} - n\Delta$, $p_{i,k+1} = \max P_{\mathsf{flat}} + n\Delta$, and $\Delta = \max P_{\mathsf{flat}} - \min P_{\mathsf{flat}}$. Thus, every point $p \in P_{\mathsf{flat}}$ can be viewed as the median of some traversal of $\widetilde{\mathcal{P}}$. Moreover, since we put the $p_{i,0}$ and $p_{i,k+1}$ points far enough out, they will essentially act as points at infinity and not affect the rest of our analysis.

Next, for $p \in P_{\mathsf{flat}}$ we define $\mathsf{cost}(p) = \min\{\mathsf{cost}(p, Q) \mid p \text{ is the median of } Q \text{ and } Q \Subset \widetilde{\mathcal{P}}\}$. Thus, if there exists $Q \Subset \mathcal{P}$ such that p is the median of Q, then $\mathsf{cost}(p) \le \mathsf{cost}(p, Q)$.

Now to compute cost and expedite our analysis, for $p \in [\min P_{\mathsf{flat}} - n\Delta, \max P_{\mathsf{flat}} + n\Delta]$, we define $L_i(p) = \min\{|p_i - p| \mid p_i \in \widetilde{P}_i \cap (-\infty, p]\}$ and $R_i(p) = \min\{|p_i - p| \mid p_i \in \widetilde{P}_i \cap [p, \infty)\}$. and recall $D_i(p) = L_i(p) - R_i(p)$. Obviously, if $p \in \widetilde{P}_i$, then $D_i(p) = L_i(p) = R_i(p) = 0$. For example, if $\widetilde{P}_i = \{p_{i,0},\ p_{i,1},\ p_{i,2},\ p_{i,3},\ p_{i,4}\}$ and $p_{i,0} < p_{i,1} < p_{i,2} < p_{i,3} < p_{i,4}$, then the plot of $L_i(p)$, $R_i(p)$ and $D_i(p)$, is shown in Figure 1.

For the sake of brevity, we now assume n is odd; adjusting a few arguments by $+1$ will adjust for the n is even case.

Consider next the following property of the D_i functions with respect to computing $\mathsf{cost}(p)$ for a point $p \in P_{i_0}$. Let $\{i_1, i_2, \cdots, i_{n-1}\} = [n] \backslash \{i_0\}$ be a permutation of uncertain points, except for i_0, so that $D_{i_1}(p) \le D_{i_2}(p) \le \cdots \le D_{i_{n-1}}(p)$. Then to minimize $\mathsf{cost}(p, Q)$, we count the uncertain points P_{i_l} using L_{i_l} if in the permutation $i_l \le (n-1)/2$ and otherwise count it on the right with R_{i_l}. This holds since for any other permutation $\{j_1, j_2, \cdots, j_{n-1}\} = [n] \backslash \{i_0\}$ we have $\sum_{l=\frac{n+1}{2}}^{n-1} D_{i_l}(p) \ge \sum_{l=\frac{n+1}{2}}^{n-1} D_{j_l}(p)$ and thus

$$\begin{aligned} &\sum_{l=1}^{\frac{n-1}{2}} L_{i_l}(p) + \sum_{l=\frac{n+1}{2}}^{n-1} R_{i_l}(p) = \sum_{l=1}^{n-1} L_{i_l}(p) - \sum_{l=\frac{n+1}{2}}^{n-1} D_{i_l}(p) \\ \le &\sum_{l=1}^{n-1} L_{j_l}(p) - \sum_{l=\frac{n+1}{2}}^{n-1} D_{j_l}(p) = \sum_{l=1}^{\frac{n-1}{2}} L_{j_l}(p) + \sum_{l=\frac{n+1}{2}}^{n-1} R_{j_l}(p). \end{aligned}$$

For $p \in P_{i_0}$, $\mathsf{cost}(p) = \frac{1}{n}\left(\sum_{l=1}^{\frac{n-1}{2}} L_{i_l}(p) + \sum_{l=\frac{n+1}{2}}^{n-1} R_{i_l}(p)\right)$ under this D_i-sorted permutation.

3.1 Computing cost

Now to compute cost for all points $p \in P_{\mathsf{flat}}$, we simply need to maintain the D_i in sorted order, and then sum the appropriate terms from L_i and R_i. Let us first examine a few facts about the complexity of these functions.

The function L_i (resp. R_i) is piecewise-linear, where the slope is always 1 (resp. -1). The breakpoints only occur at $x = p_{i,j}$ for each $p_{i,j} \in P_i$. Hence, they each have complexity $\Theta(k)$ for all $i \in [n]$. The structure of L_i and R_i implies that D_i is also piecewise-linear, where the slope is always 2 and has breakpoints for each $p_{i,j} \in P_i$. Each linear component attains a value $D_i(x) = 0$ when x is the midpoint between two $p_{i,j}, p_{i,j'} \in P_i$ which are consecutive in the sorted order of P_i.

The fact that all D_i have slope 2 at all non-discontinuous points, and these discontinuous points only occur at P_i, implies that the sorted order of the D_i functions does not change in between points of P_{flat}. Moreover, at one of these points of discontinuity $x \in P_{\mathsf{flat}}$, the ordering between D_is only changes for uncertain points $D_{i'}$ such that there exists a possible location $p_{i',j} \in P_{i'}$ such that $x = p_{i',j}$. This implies that to maintain the sorted order of D_i for any x, as we increase the value of x, we only need to update this order at the nk points in P_{flat} with respect to $D_{i'}$ for which there exists $p_{i',j} \in P_{i'}$ with $p_{i',j} = x$. This takes $O(\log(nk))$ time per update using a balanced BST, and thus $O(nk \log(nk))$ time to define $\mathsf{cost}(x)$ for all values $x \in \mathbb{R}$[1]. To compute $\mathsf{cost}(x)$, we also require the values of L_i (or R_i); these can be constructed independently for each $i \in [n]$ in $O(k)$ time after sorting, and in $O(nk \log k)$ time overall.[1] Ultimately, we arrive at the following theorem.

▶ **Theorem 2.** *Consider a set of n uncertain points $\mathcal{P}$ with k possible locations each. We can compute $\mathsf{cost}(x)$ for all $x \in \mathbb{R}$ such that $x = p_{i,j}$ for some $p_{i,j} \in P_{\mathsf{flat}}$ in $O(nk \log(nk))$ time.*

3.2 Building the ε-support T and bounding its size

We next show that there always exists an ε-support T and it has a size $|T| = O(\frac{k}{\varepsilon})$.

▶ **Theorem 3.** *Given a set of n uncertain points $\mathcal{P} = \{P_1, \cdots, P_n\}$, where $P_i = \{p_{i,1}, \cdots, p_{i,k}\} \subset \mathbb{R}$, and $\varepsilon \in (0, 1]$ we can construct an ε-support T that has a size $|T| = O(\frac{k}{\varepsilon})$.*

Proof. We first sort P_{flat} in ascending order, scan $P_{\mathsf{flat}} = \{p_1, \cdots, p_{nk}\}$ from left to right and choose one point from P_{flat} every $\lfloor \frac{n}{3} \rfloor$ points, and then put the chosen point into T. Now, suppose p is the median of some traversal $Q \in \mathcal{P}$ and $\mathsf{cost}(p) = \mathsf{cost}(p, Q)$. If $p \notin T$, then there are two consecutive points t, t' in T such that $t < p < t'$. On either side of p there are at least $\lfloor \frac{n}{2} \rfloor$ points in Q, so without loss of generality, we assume $|p - t'| \geq \frac{1}{2}|t - t'|$. Since $|[p, \infty) \cap Q| \geq \frac{n}{2}$ and there are at most $\lfloor \frac{n}{3} \rfloor$ points in $[p, t']$, we have $|(t', \infty) \cap Q| \geq \frac{n}{2} - \lfloor \frac{n}{3} \rfloor \geq \frac{n}{6}$, which implies

$$\begin{aligned}\mathsf{cost}(p) =& \mathsf{cost}(p, Q) \geq \frac{1}{n} \sum\nolimits_{q \in (t', \infty) \cap Q} |q - p| \geq \frac{1}{n} \sum\nolimits_{q \in (t', \infty) \cap Q} |t' - p| \\ \geq& \frac{1}{n} \frac{n}{6} |t' - p| = \frac{1}{6} |t' - p| \geq \frac{1}{12} |t - t'|. \end{aligned} \tag{1}$$

For any fixed $\varepsilon \in (0, 1]$, and two consecutive points t, t' $(t < t')$ in T, we put $x_1, \cdots, x_{\lceil \frac{12}{\varepsilon} \rceil - 1}$ into T where $x_i = t + \frac{|t - t'| i}{\lceil \frac{12}{\varepsilon} \rceil}$ for $1 \leq i \leq \lceil \frac{12}{\varepsilon} \rceil - 1$. So, for the median $p \in (t, t')$, there exists $x_i \in T$ s.t. $|p - x_i| \leq \frac{\varepsilon}{12} |t - t'|$, and from (1), we know $|p - x_i| \leq \varepsilon \mathsf{cost}(p)$. In total we put $O(\frac{k}{\varepsilon})$ points into T; thus the proof is completed. ◀

[1] When multiple distinct $p_{i,j}$ coincide at a point x, then more care may be required to compute $\mathsf{cost}(x)$ (depending on the specifics of how the median is defined in these boundary cases). Specifically, we may not want to set $L_i(x) = 0$, instead it may be better to use the value $R_i(x)$ even if $R_i(x) = \alpha > 0$. This is the case when $\alpha < R_{i'}(x) - L_{i'}(x)$ for some other uncertain point $P_{i'}$ (then we say P_i is on the right, and P_i is on the left). This can be resolved by either tweaking the definition of median for these cases, or sorting all $D_i(x)$ for uncertain points P_i with some $p_{i,j} = x$, and some bookkeeping.

▶ Remark. The above construction results in an ε-support T of size $O(k/\varepsilon)$, but does not restrict that $T \subset P_{\mathsf{flat}}$. We can enforce this restriction by for each x placed in T to choose the single nearest point $p \in P_{\mathsf{flat}}$ to replace it in T. This results in an (2ε)-support, which can be made an ε-support by instead adding $\lceil \frac{24}{\varepsilon} \rceil - 1$ points between each pair (t, t'), without affecting the asymptotic time bound.

▶ Remark. We can construct a sequence of uncertain data $\{\mathcal{P}(n,k)\}$ such that, for each uncertain data $\mathcal{P}(n,k)$, the optimal ε-support T has a size $\Omega(\frac{k}{\varepsilon})$. For example, for $\varepsilon = \frac{1}{3}, \frac{1}{5}, \frac{1}{7}, \cdots$, we define $n = \frac{1}{\varepsilon}$, and $p_{i,j} = (j-1)n + i$ for $i \in [n]$ and $j \in [k]$. Then, for any median $p \in P_{\mathsf{flat}}$, we have $\varepsilon \mathsf{cost}(p) = \frac{2}{n^2} \sum_{i=1}^{\frac{n-1}{2}} i = \frac{n^2-1}{4n^2} < \frac{1}{4}$, hence covering no other points, which implies $|T| = \Omega(nk) = \Omega(\frac{k}{\varepsilon})$.

We can construct the minimal size ε-support T in $O(nk \log(nk))$ time by sorting, and greedily adding the smallest point not yet covered each step. This yields the slightly stronger corollary of Theorem 3.

▶ **Corollary 4.** *Consider a set of n uncertain points $\mathcal{P} = \{P_1, \cdots, P_n\}$, where $P_i = \{p_{i,1}, \cdots, p_{i,k}\} \subset \mathbb{R}$, and $\varepsilon \in (0, 1]$. We can construct an ε-support T in $O(nk \log(nk))$ time which has the minimal size for any ε-support, and $|T| = O(\frac{k}{\varepsilon})$.*

There are multiple ways to generalize the notion of a median to higher dimensions [5]. We focus on two variants: the Tukey median and the geometric median. We start with generalizing the notion of an ε-support to a Tukey median since it more directly follows from the techniques in Theorem 3, and then address the geometric median.

3.3 An ε-Support for the Tukey median

A closely related concept to the Tukey median is a *centerpoint*, which is a point p such that $\mathsf{depth}_Q(p) \geq \frac{1}{d+1}|Q|$. Since for any finite set $Q \in \mathbb{R}^d$ its centerpoint always exists, a Tukey median must be a centerpoint. This means if p is the Tukey median of Q, then for any closed half space containing p, it contains at least $\frac{1}{d+1}|Q|$ points of Q. Using this property, we can prove the following theorem.

▶ **Theorem 5.** *Given a set of n uncertain points $\mathcal{P} = \{P_1, \cdots, P_n\}$, where $P_i = \{p_{i,1}, \cdots, p_{i,k}\} \subset \mathbb{R}^2$, and $\varepsilon \in (0, 1]$, we can construct an ε-support T for the Tukey median on $\mathcal{P}$ that has a size $|T| = O(\frac{k^2}{\varepsilon^2})$.*

Proof. Suppose the projections of P_{flat} on x-axis and y-axis are X and Y respectively. We sort all points in X and choose one point from X every $\lfloor \frac{n}{4} \rfloor$ points, and then put the chosen points into a set X_T. For each point $x \in X_T$ we draw a line through $(x, 0)$ parallel to y-axis. Similarly, we sort all points in Y and choose one point every $\lfloor \frac{n}{4} \rfloor$ points, and put the chosen points into Y_T. For each point $y \in Y_T$ we draw a line through $(0, y)$ parallel to x-axis.

Now, suppose p with coordinates (x_p, y_p) is the Tukey median of some traversal $Q \Subset \mathcal{P}$ and $\mathsf{cost}(p, Q) = \frac{1}{n} \sum_{q \in Q} \|q - p\|$. If $x_p \notin X_T$ and $y_p \notin Y_T$, then there are $x, x' \in X_T$ and $y, y' \in Y_T$ such that $x < x_p < x'$ and $y < y_p < y'$, as shown in Figure 2(a).

Without loss of generality, we assume $|x_p - x| \geq \frac{1}{2}|x' - x|$ and $|y_p - y| \geq \frac{1}{2}|y - y'|$. Since p is the Tukey median of Q, we have $|Q \cap (-\infty, \infty) \times (-\infty, y_p]| \geq \frac{n}{3}$ where $(-\infty, \infty) \times (-\infty, y_p] = \{(x, y) \in \mathbb{R}^2 | \ y \leq y_p\}$. Recall there are at most $\lfloor \frac{n}{4} \rfloor$ points of P_{flat} in $(-\infty, \infty) \times [y_p, y]$, which implies $|Q \cap (-\infty, \infty) \times (-\infty, y)| \geq \frac{n}{3} - \lfloor \frac{n}{4} \rfloor \geq \frac{n}{12}$. So, we have

$$\mathsf{cost}(p, Q) \geq \frac{1}{n} \sum_{q \in Q \cap (-\infty, \infty) \times (-\infty, y)} \|q - p\| \geq \frac{1}{n} \frac{n}{12} |y - y_p| \geq \frac{1}{24} |y - y'|.$$

Using a symmetric argument, we can obtain $\mathsf{cost}(p,Q) \geq \frac{1}{24}|x-x'|$.

For any fixed $\varepsilon \in (0,1]$, and any two consecutive points x, x' in X_T we put $x_1, \cdots, x_{\lceil \frac{48}{\varepsilon} \rceil - 1}$ into X_T where $x_i = x + \frac{|x-x'|i}{\lceil \frac{48}{\varepsilon} \rceil}$. Also, for any two consecutive point y, y' in Y_T, we put $y_1, \cdots, y_{\lceil \frac{48}{\varepsilon} \rceil - 1}$ into Y_T where $y_i = y + \frac{|y-y'|i}{\lceil \frac{48}{\varepsilon} \rceil}$. So, for the Tukey median $p \in (x,x') \times (y,y')$, there exist $x_i \in X_T$ and $y_j \in Y_T$ such that $|x_p - x_i| \leq \frac{\varepsilon}{48}|x-x'|$ and $|y_p - y_j| \leq \frac{\varepsilon}{48}|y-y'|$. Since we have shown that $\frac{1}{24}|x-x'|$ and $\frac{1}{24}|y-y'|$ are lower bounds for $\mathsf{cost}(p,Q)$, we obtain

$$\begin{aligned}\|(x_p,y_p)-(x_i,y_j)\| \leq & |x_p - x_i| + |y_p - y_j| \leq \frac{\varepsilon}{48}(|x-x'| + |y-y'|) \\ \leq & \frac{\varepsilon}{48}(24\mathsf{cost}(p,Q) + 24\mathsf{cost}(p,Q)) = \varepsilon \mathsf{cost}(p,Q).\end{aligned}$$

Finally, we define T as $T := X_T \times Y_T$. Then for any $Q \Subset \mathcal{P}$, if p is the Tukey median of Q, there exists $t \in T$ such that $\|t-p\| \leq \varepsilon \mathsf{cost}(p,Q)$. Thus, T is an ε-support for the Tukey median on $\mathcal{P}$. Moreover, since $|X_T| = O(\frac{k}{\varepsilon})$ and $|Y_T| = O(\frac{k}{\varepsilon})$, we have $|T| = O(\frac{k^2}{\varepsilon^2})$. ◀

In a straight-forward extension, we can generalize the result of Theorem 5 to d dimensions.

▶ **Theorem 6.** *Given a set of n uncertain points $\mathcal{P} = \{P_1, \cdots, P_n\}$, where $P_i = \{p_{i,1}, \cdots, p_{i,k}\} \subset \mathbb{R}^d$, and $\varepsilon \in (0,1]$, we can construct an ε-support T for the Tukey median on $\mathcal{P}$ that has a size $|T| = O((2d(d+1)(d+2)^2 \frac{k}{\varepsilon})^d)$.*

3.4 An ε-support for the geometric median

Unlike the Tukey median, there does not exist a constant $C > 0$ such that: for any geometric median p of point set $Q \subset \mathbb{R}^d$, any closed halfspace containing p contains at least $\frac{1}{C}|Q|$ points of Q. For example, suppose in $\mathbb{R}^2$ there are $2n+1$ points on x-axis with the median point at the origin; this point is also the geometric median. If we move this point upward along the y direction, then the geometric median also moves upwards. However, for the line through the new geometric median and parallel to the x-axis, all $2n$ other points are under this line.

Hence, we need a new idea to adapt the method in Theorem 6 for the geometric median in $\mathbb{R}^d$. We first consider the geometric median in $\mathbb{R}^2$. We show we can find *some* line through it, such that on both sides of this line there are at least $\frac{n}{8}$ points.

▶ **Lemma 7.** *Suppose p is the geometric median of $Q \subset \mathbb{R}^2$ with size $|Q| = n$. There is a line ℓ through p so both closed half planes with ℓ as boundary contain at least $\frac{n}{8}$ points of Q.*

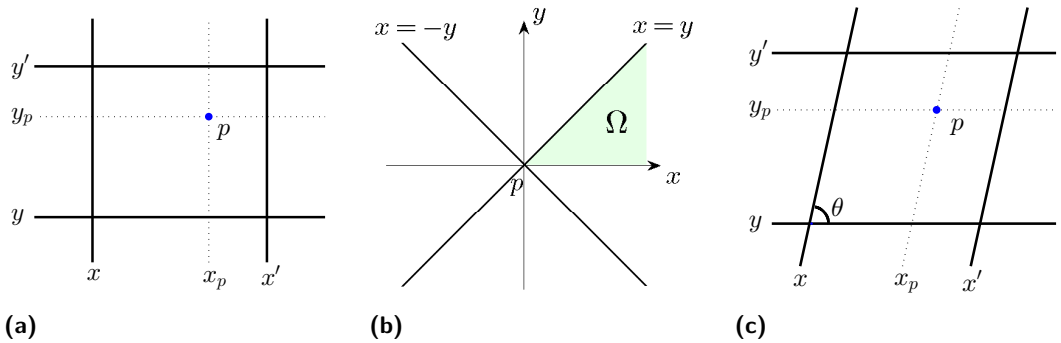


Figure 2 (a) Tukey median p is in a grid cell formed by x, x' and y, y'. (b) The plane is decomposed into 8 regions with the same shape. (c) Geometric median p is in an oblique grid cell formed by x, x' and y, y'.

Proof. We first build a rectangular coordinate system at the point p, which means p is the origin with coordinates $(x_p, y_p) = (0, 0)$. Then we use the x-axis, y-axis and lines $x = y$, $x = -y$ to decompose the plane into eight regions, as shown in Figure 2(b). Since all these eight regions have the same shape, without loss of generality, we can assume $\Omega = \{(x, y) \in \mathbb{R}^2 | \ x \geq y \geq 0\}$ contains the most points of Q. Then $|\Omega \cap Q| \geq \frac{n}{8}$, otherwise $n = |Q| = |\mathbb{R}^2 \cap Q| \leq 8|\Omega \cap Q| < n$, which is a contradiction.

If $|Q \cap \{p\}| \geq \frac{n}{8}$, i.e., the multiset Q contains p at least $\frac{n}{8}$ times, then obviously this proposition is correct. So, we only need to consider the case $|Q \cap \{p\}| < \frac{n}{8}$. We introduce notations $\widetilde{\Omega} = \Omega \setminus \{p\}$ and $\Omega^o = \Omega \setminus \partial\Omega$, and denote the coordinates of any $q \in Q$ as $q = (x_q, y_q)$. From a property of the geometric median (proven in the full version [8]) we know $\sum_{q \in Q \setminus \{p\}} \frac{x_q - x_p}{\|q-p\|} \leq |Q \cap \{p\}|$. Since p is the origin and $Q \setminus \{p\} = (Q \cap \widetilde{\Omega}) \cup (Q \setminus \Omega)$, $|Q \cap \{p\}| < \frac{n}{8}$ implies $\sum_{q \in Q \cap \widetilde{\Omega}} \frac{x_q}{\|q\|} + \sum_{q \in Q \setminus \Omega} \frac{x_q}{\|q\|} < \frac{n}{8}$. From $\frac{x_q}{\|q\|} = \frac{x_q}{\sqrt{x_q^2 + y_q^2}} \geq \frac{1}{\sqrt{2}}$, $\forall q \in \widetilde{\Omega}$ we obtain

$$|Q \cap \widetilde{\Omega}| \frac{1}{\sqrt{2}} \leq \sum_{q \in Q \cap \widetilde{\Omega}} \frac{x_q}{\|q\|} < \frac{n}{8} - \sum_{q \in Q \setminus \Omega} \frac{x_q}{\|q\|} \leq \frac{n}{8} + |Q \setminus \Omega| \leq \frac{n}{8} + (n - |Q \cap \widetilde{\Omega}|)$$

which implies there are not too many points in $\widetilde{\Omega}$,

$$|Q \cap \widetilde{\Omega}| < \frac{\sqrt{2}n}{(1 + \sqrt{2})} \cdot \frac{9}{8} < 0.66n.$$

Now, we define the two pairs of halfspaces which share a boundary with $\widetilde{\Omega}$: $H_1^+ = \{(x, y) \in \mathbb{R}^2 | \ y \geq 0\}$, $H_1^- = \{(x, y) \in \mathbb{R}^2 | \ y \leq 0\}$ and $H_2^+ = \{(x, y) \in \mathbb{R}^2 | \ x - y \geq 0\}$, $H_2^- = \{(x, y) \in \mathbb{R}^2 | \ x - y \leq 0\}$. We assert either $|H_1^+ \cap Q| \geq \frac{n}{8}$ and $|H_1^- \cap Q| \geq \frac{n}{8}$, or $|H_2^+ \cap Q| \geq \frac{n}{8}$ and $|H_2^- \cap Q| \geq \frac{n}{8}$. Otherwise, since $|Q \cap \Omega| \geq \frac{n}{8}$ and $\Omega \subset H_1^+ \cap H_2^+$, we have $|H_1^- \cap Q| < \frac{n}{8}$ and $|H_2^- \cap Q| < \frac{n}{8}$. From $H_1^- \cup H_2^- \cup \Omega^o = \mathbb{R}^2$ we have

$$\begin{aligned} n =& |Q| = |\mathbb{R}^2 \cap Q| = |(H_1^- \cup H_2^- \cup \Omega^o) \cap Q| \leq |H_1^- \cap Q| + |H_2^- \cap Q| + |\Omega^o \cap Q| \\ \leq& |H_1^- \cap Q| + |H_2^- \cap Q| + |\widetilde{\Omega} \cap Q| \leq \frac{n}{8} + \frac{n}{8} + 0.66n < n, \end{aligned}$$

which is a contradiction. Therefore, among lines $\ell_1 : y = 0$ and $\ell_2 : x - y = 0$, which both go through p, one of them has at least $n/8$ points from Q on both sides. ◀

▶ **Theorem 8.** *Given a set of n uncertain points $\mathcal{P} = \{P_1, \cdots, P_n\}$, where $P_i = \{p_{i,1}, \cdots, p_{i,k}\} \subset \mathbb{R}^2$, and $\varepsilon \in (0, 1]$, we can construct an ε-support T for the geometric median on $\mathcal{P}$ that has a size $|T| = O(\frac{k^2}{\varepsilon^2})$.*

Proof. The idea to prove this theorem is to use several oblique coordinate systems. We consider an oblique coordinate system, the angle between x-axis and y-axis is $\theta \in (0, \frac{\pi}{2}]$, and use the technique in Theorem 5 to generate a grid. More precisely, we project P_{flat} onto the x-axis along the y-axis of the oblique coordinate system to obtain a set X, sort all points in X, and choose one point from X every $\lfloor \frac{n}{9} \rfloor$ points to form a set X_T. Then we use the same method to generate Y and Y_T projecting along the x-axis in the oblique coordinate system. For each point $x \in X_T$ we draw a line through $(x, 0)$ parallel to the (oblique) y-axis, and for each point $y \in Y_T$ we draw a line through $(0, y)$ parallel to the (oblique) x-axis.

Let p with coordinates (x_p, y_p) be the geometric median of some traversal $Q \Subset \mathcal{P}$ and $\mathsf{cost}(p, Q) = \frac{1}{n} \sum_{q \in Q} \|q - p\|$. If $x_p \notin X_T$ and $y_p \notin Y_T$, then there are $x, x' \in X_T$ and $y, y' \in Y_T$ such that $x_p \in (x, x')$ and $y_p \in (y, y')$, as shown in Figure 2(c).

If we have the condition:

$$\begin{aligned} &|Q \cap (-\infty, \infty) \times (-\infty, y_p]| \geq \frac{n}{8}, \quad |Q \cap (-\infty, \infty) \times [y_p, \infty)| \geq \frac{n}{8}, \\ &|Q \cap (-\infty, x_p] \times (-\infty, \infty)| \geq \frac{n}{8}, \quad |Q \cap [x_p, \infty) \times (-\infty, \infty)| \geq \frac{n}{8}, \end{aligned} \tag{2}$$

then we can make the following computation.

Without loss of generality, we assume $|x_p - x| \geq \frac{1}{2}|x' - x|$ and $|y_p - y| \geq \frac{1}{2}|y - y'|$. There are at most $\lfloor \frac{n}{9} \rfloor$ points of P_{flat} in $(-\infty, \infty) \times [y_p, y]$, which implies $|Q \cap (-\infty, \infty) \times (-\infty, y)| \geq \frac{n}{8} - \lfloor \frac{n}{9} \rfloor \geq \frac{n}{72}$. So, we have

$$\mathsf{cost}(p, Q) \geq \frac{1}{n} \sum\nolimits_{q \in Q \cap (-\infty, \infty) \times (-\infty, y)} \|q - p\| \geq \frac{1}{n} \frac{n}{72} |y - y_p| \geq \frac{\sin(\theta)}{144} |y - y'|.$$

Similarly, we can prove $\mathsf{cost}(p, Q) \geq \frac{\sin(\theta)}{144}|x - x'|$.

For any fixed $\varepsilon \in (0, 1]$, and any two consecutive points x, x' in X_T we put $x_1, \cdots, x_{\lceil \frac{288}{\varepsilon \sin(\theta)} \rceil - 1}$ into X_T where $x_i = x + \frac{|x - x'| i}{\lceil \frac{288}{\varepsilon \sin(\theta)} \rceil}$. Also, for any two consecutive point y, y' in Y_T, we put $y_1, \cdots, y_{\lceil \frac{288}{\varepsilon \sin(\theta)} \rceil - 1}$ into Y_T where $y_i = y + \frac{|y - y'| i}{\lceil \frac{288}{\varepsilon \sin(\theta)} \rceil}$. So, for the L_1 median $p \in (x, x') \times (y, y')$, there exist $x_i \in X_T$ and $y_j \in Y_T$ such that $|x_p - x_i| \leq \frac{\varepsilon \sin(\theta)}{288}|x - x'|$ and $|y_p - y_j| \leq \frac{\varepsilon \sin(\theta)}{288}|y - y'|$. Since we have shown that both $\frac{\sin(\theta)}{144}|x - x'|$ and $\frac{\sin(\theta)}{144}|y - y'|$ are lower bounds for $\mathsf{cost}(p, Q)$, using the distance formula in an oblique coordinate system, we have

$$\begin{aligned} \|(x_p, y_p) - (x_i, y_j)\| \leq& ((x_p - x_i)^2 + (y_p - y_j)^2 + 2(x_p - x_i)(y_i - y_p)\cos(\theta))^{\frac{1}{2}} \\ \leq& ((x_p - x_i)^2 + (y_p - y_j)^2 + 2|x_p - x_i||y_i - y_p|)^{\frac{1}{2}} \\ =& |x_p - x_i| + |y_p - y_j| \leq \frac{\varepsilon \sin(\theta)}{288}(|x - x'| + |y - y'|) \\ \leq& \frac{\varepsilon \sin(\theta)}{288}\left(\frac{144}{\sin(\theta)}\mathsf{cost}(p, Q) + \frac{144}{\sin(\theta)}\mathsf{cost}(p, Q)\right) = \varepsilon \mathsf{cost}(p, Q). \end{aligned}$$

Therefore, if all k^n geometric medians of traversals satisfy (2) and $\theta \in (0, \frac{\pi}{2}]$ is a constant then $T = X_T \times Y_T$ is an ε-support of size $O\left(\frac{k^2}{(\sin(\theta)\varepsilon)^2}\right)$ for the geometric median on $\mathcal{P}$.

Although we cannot find an oblique coordinate system to make (2) hold for all k^n medians, we can use several oblique coordinate systems. Using the result of Lemma 7, for any geometric median of n points Q, we know there exists a line ℓ through p and parallel to a line in $\{\ell_1 : y = 0,\ \ell_2 : x - y = 0,\ \ell_3 : x = 0,\ \ell_4 : x + y = 0\}$, such that in both sides of this line, there are at least $\frac{n}{8}$ points of Q. Since we did not make any assumption on the distribution of points in Q, if we rotate $\ell_1, \ell_2, \ell_3, \ell_4$ anticlockwise by $\frac{\pi}{8}$ around the origin, we can obtain four lines $\ell'_1, \ell'_2, \ell'_3, \ell'_4$, and there exists a line ℓ' through p and parallel to a line in $\{\ell'_1, \ell'_2, \ell'_3, \ell'_4\}$, such that on both sides of this line, there are at least $\frac{n}{8}$ points of Q. The angle between ℓ and ℓ' is at least $\frac{\pi}{8}$.

Therefore, given $\mathcal{L} = \{\ell_1, \ell_2, \ell_3, \ell_4\}$ and $\mathcal{L}' = \{\ell'_1, \ell'_2, \ell'_3, \ell'_4\}$, for each pair $(\ell, \ell') \in \mathcal{L} \times \mathcal{L}'$, we take ℓ and ℓ' as x-axis and y-axis respectively to build an oblique coordinate system, and then use the above method to compute a set $T(\ell, \ell')$. Since for any geometric median p there must be an oblique coordinate system based on some $(\ell, \ell') \in \mathcal{L} \times \mathcal{L}'$ to make (2) hold for p, we can take $T = \cup_{\ell \in \mathcal{L}, \ell' \in \mathcal{L}'} T(\ell, \ell')$ as an ε-support for geometric median on $\mathcal{P}$, and the size of T is $|T| = O\left(16\frac{k^2}{(\sin(\frac{\pi}{8})\varepsilon)^2}\right) = O\left(\frac{k^2}{\varepsilon^2}\right)$. ◀

The result of Theorem 8 can be generalized to $\mathbb{R}^d$ and details are in the full version [8].

3.5 Assigning a weight to T in $\mathbb{R}^1$

Here we provide an algorithm to assign a weight to T in $\mathbb{R}^1$, which approximates the probability distribution of median. For T in $\mathbb{R}^d$, we provide a randomized algorithm in Section 4.1.

Define the weight of $p_{i,j} \in P_{\mathsf{flat}}$ as $w(p_{i,j}) = \frac{1}{k^n}|\{Q \Subset \mathcal{P} \mid p_{i,j} \text{ is the median of Q}\}|$, the probability it is the median. Suppose T is constructed by our greedy algorithm for $\mathbb{R}^1$. For $p_{i,j} \in P_{\mathsf{flat}}$, we introduce a map $f_T : P_{\mathsf{flat}} \to T$,

$$f_T(p_{i,j}) = \arg\min\{|x - p_{i,j}| \mid x \in T,\ |x - p_{i,j}| \leq \varepsilon \mathsf{cost}(p_{i,j})\},$$

where $\mathsf{cost}(p_{i,j}) = \min\{\mathsf{cost}(p_{i,j}, Q) \mid p_{i,j} \text{ is the median of } Q \text{ and } Q \Subset \mathcal{P}\}$.

Intuitively, this maps each $p_{i,j} \in P_{\mathsf{flat}}$ onto the closest point $x \in T$, unless it violates the ε-approximation property which another further point satisfies.

Now for each $x \in T$, define weight of x as $\hat{w}(x) = \sum_{\{p_{i,j} \in P_{\mathsf{flat}} | f_T(p_{i,j}) = x\}} w(p_{i,j})$. So we first compute the weight of each point in P_{flat} and then obtain the weight of points in T in another linear sweep. Our ability to calculate the weights w for each point in P_{flat} is summarized in the next lemma. The algorithm, explained within the proof, is a dynamic program that expands a specific polynomial similar to Li *et.al.* [14], where in the final state, the coefficients correspond with the probability of each point being the median.

▶ **Lemma 9.** *We can output $w(p_{i,j})$ for all points in P_{flat} in $\mathbb{R}^1$ in $O(n^2k)$ time.*

Proof. For any $p_{i_0} \in P_{i_0}$, we define the following terms to count the number of points to the left (l_j) or right (r_j) of it in the jth uncertain point (excluding P_{i_0}):

$$l_j = \begin{cases} |\{p \in P_j \mid p \leq p_{i_0}\}| & \text{if } 1 \leq j \leq i_0 - 1 \\ |\{p \in P_{j+1} \mid p \leq p_{i_0}\}| & \text{if } i_0 \leq j \leq n-1 \end{cases}, \quad r_j = \begin{cases} |\{p \in P_j \mid p \geq p_{i_0}\}| & \text{if } 1 \leq j \leq i_0 - 1 \\ |\{p \in P_{j+1} \mid p \geq p_{i_0}\}| & \text{if } i_0 \leq j \leq n-1 \end{cases}.$$

Then, if n is odd, we can write the weight of p_{i_0} as

$$w(p_{i_0}) = \frac{1}{k^n} \sum_{\substack{S_1 \cap S_2 = \emptyset \\ S_1 \cup S_2 = \{1,\cdots,n-1\}}} (l_{i_1} \cdot l_{i_2} \cdot \ldots \cdot l_{i_{\frac{n-1}{2}}} \cdot r_{j_1} \cdot r_{j_2} \cdot \ldots \cdot r_{j_{\frac{n-1}{2}}}),$$

where $S_1 = \{i_1, i_2, \cdots, i_{\frac{n-1}{2}}\}$ and $S_2 = \{j_1, j_2, \cdots, j_{\frac{n-1}{2}}\}$. This sums over all partitions S_1, S_2 of uncertain points on the left or right of p_{i_0} for which it is the median, and each term is the product of ways each uncertain point can be on the appropriate side. We define $w(p_{i_0})$ similarly when n is even, then the last index of S_2 is $j_{\frac{n}{2}}$.

We next describe the algorithm for n odd; the case for n even is similar. To compute $\sum_{\substack{S_1 \cap S_2 = \emptyset \\ S_1 \cup S_2 = \{1,\cdots,n-1\}}} (l_{i_1} \cdot l_{i_2} \cdot \ldots \cdot l_{i_{\frac{n-1}{2}}} \cdot r_{j_1} \cdot r_{j_2} \cdot \ldots \cdot r_{j_{\frac{n-1}{2}}})$, we consider the following polynomial:

$$(l_1 x + r_1)(l_2 x + r_2) \cdots (l_{n-1} x + r_{n-1}), \tag{3}$$

where $\sum_{\substack{S_1 \cap S_2 = \emptyset \\ S_1 \cup S_2 = \{1,\cdots,n-1\}}} (l_{i_1} \cdot l_{i_2} \cdot \ldots \cdot l_{i_{\frac{n-1}{2}}} \cdot r_{j_1} \cdot r_{j_2} \cdot \ldots \cdot r_{j_{\frac{n-1}{2}}})$ is the coefficient of $x^{\frac{n-1}{2}}$. We define $\rho_{i,j}$ $(1 \leq i \leq n-1, 0 \leq j \leq i)$ as the coefficient of x^j in the polynomial $(l_1 x + r_1) \cdots (l_i x + r_i)$ and then it is easy to check $\rho_{i,j} = l_i \rho_{i-1,j-1} + r_i \rho_{i-1,j}$. Thus we can use dynamic programming to compute $\rho_{n-1,0}, \rho_{n-1,1}, \cdots, \rho_{n-1,n-1}$, as shown in Algorithm 1.

Thus Algorithm 1 computes the weight $\frac{1}{k^n} w(p_{i_0}) = \rho_{n-1,\frac{n-1}{2}}$ for a single $p_{i_0} \in P_{\mathsf{flat}}$. Next we show, we can reuse much of the structure to compute the weight for another point; this will ultimately shave a factor n off of running Algorithm 1 nk times.

Algorithm 1 Compute $\rho_{n-1,0}, \rho_{n-1,1}, \cdots, \rho_{n-1,n-1}$

Let $\rho_{1,0} = r_1, \rho_{1,1} = l_1, \rho_{1,2} = 0$.
for $i = 2$ to $n-1$ **do**
  for $j = 0$ to i **do**
    $\rho_{i,j} = l_i \rho_{i-1,j-1} + r_i \rho_{i-1,j}$
  $\rho_{i,i+1} = 0$
return $\rho_{n-1,0}, \rho_{n-1,1}, \cdots, \rho_{n-1,n-1}$.

Suppose for $p_{i_0} \in P_{i_0}$ we have obtained $\rho_{n-1,0}, \rho_{n-1,1}, \ldots, \rho_{n-1,n-1}$ by Algorithm 1, and then we consider $p_{i'_0} = \min\{p \in P_{\mathsf{flat}} \setminus P_{i_0} \mid p \geq p_{i_0}\}$. We assume $p_{i'_0} \in P_{i'_0}$, and if $i'_0 < i_0$, we construct a polynomial

$$(l_1 x + r_1) \cdots (l_{i'_0 - 1} x + r_{i'_0 - 1})(\tilde{l}_{i'_0} x + \tilde{r}_{i'_0})(l_{i'_0+1} x + r_{i'_0+1}) \cdots (l_{n-1} x + r_{n-1}) \tag{4}$$

and if $i'_0 > i_0$, we construct a polynomial

$$(l_1 x + r_1) \cdots (l_{i'_0 - 2} x + r_{i'_0 - 2})(\tilde{l}_{i'_0 - 1} x + \tilde{r}_{i'_0 - 1})(l_{i'_0} x + r_{i'_0}) \cdots (l_{n-1} x + r_{n-1}) \tag{5}$$

where $\tilde{l}_{i'_0} = \tilde{l}_{i'_0 - 1} = |\{p \in P_{i_0} \mid p \leq p_{i'_0}\}|$ and $\tilde{r}_{i'_0} = \tilde{r}_{i'_0 - 1} = |\{p \in P_{i_0} \mid p \geq p_{i'_0}\}|$.

Since (3) and (4) have only one different factor, we obtain the coefficients of (4) from the coefficients of (3) in $O(n)$ time. We recover the coefficients of $(l_1 x + r_1) \cdots (l_{i'-1} x + r_{i'-1})(l_{i'_0+1} x + r_{i'_0+1}) \cdots (l_{n-1} x + r_{n-1})$ from $\rho_{n-1,0}, \rho_{n-1,1}, \cdots, \rho_{n-1,n-1}$, and then use these coefficients to compute the coefficients of (4). Similarly, if $i'_0 > i_0$, we obtain the coefficients of (5) from the coefficients of (3). Therefore, we can use $O(n^2)$ time to compute the weight of the first point in P_{flat} and then use $O(n)$ time to compute the weight of each other point. The whole time is $O(n^2) + nkO(n) = O(n^2 k)$. ◀

▶ **Corollary 10.** *We can assign $\hat{w}(x)$ to each $x \in T$ in $\mathbb{R}^1$ in $O(n^2 k)$ time.*

4 A randomized algorithm to construct a covering set

In this section we describe a much more general randomized algorithm for robust estimators on uncertain data. It constructs an approximate covering set of the support of the distribution of the estimator, and estimates the weight at the same time. The support of the distribution is not as precise compared to the techniques in the previous section in that the new technique may fail to cover regions with small probability of containing the estimator.

Suppose $\mathcal{P} = \{P_1, P_2, \cdots, P_n\}$ is a set of uncertain data, where for $i \in [n]$, $P_i = \{p_{i,1}, p_{i,2}, \cdots, p_{i,k}\} \subseteq \mathcal{X}$ for some domain $\mathcal{X}$. An estimator $E : \{Q \mid Q \Subset \mathcal{P}\} \mapsto Y$ maps $Q \Subset \mathcal{P}$ to a metric space (Y, φ). Let $B(y, r) = \{y' \in Y \mid \varphi(y, y') \leq r\}$ be a ball of radius r in that metric space. We denote ν as the VC-dimension of the range space $(Y, \mathcal{R})$ induced by these balls, with $\mathcal{R} = \{B(y, r) \mid y \in Y, r \geq 0\}$.

We now analyze the simple algorithm which randomly instantiates traversals $Q \Subset \mathcal{P}$, and constructors their estimators $z = E(Q)$. Repeating this N times builds a domain $T = \{z_1, z_2, \ldots, z_N\}$ each with weight $w(z_i) = 1/N$.

▶ **Theorem 11.** *For $\varepsilon > 0$ and $\delta \in (0, 1)$, set $N = O((1/\varepsilon^2)(\nu + \log(1/\delta)))$. Then, with probability at least $1 - \delta$, for any $B \in \mathcal{R}$ we have $\left|\sum_{z \in T \cap B} w(z) - \Pr_{Q \Subset \mathcal{P}}[E(Q) \in B]\right| \leq \varepsilon$.*

Proof. Let T^* be the true support of $E(Q)$ where $Q \Subset P$, and let $w^* : T^* \to \mathbb{R}^+$ be the true probability distribution defined on T^*; e.g., for discrete T^*, then for any $z' \in T^*$,

$w^*(z') = \Pr_{Q \in P}[E(Q) = z']$. Then each random z generated is a random draw from w^*. Hence for a range space with bounded VC-dimension [21] ν, we can apply the sampling bound [15] for ε-approximations of these range spaces to prove our claim. ◀

In Theorem 11, for $z_i \in T$, if we choose $B = B(z_i, r) \in \mathcal{R}$ with r small enough such that $T \cap B$ only contains z_i, then we obtain the following.

▶ **Corollary 12.** *For $\varepsilon > 0$ and $\delta \in (0,1)$, set $N = O((1/\varepsilon^2)(\nu + \log(1/\delta)))$. Then, with probability at least $1 - \delta$, for any $z \in Y$ we have $|w(z) - \Pr_{Q \in \mathcal{P}}[E(Q) = z]| \leq \varepsilon$.*

▶ Remark. We can typically define a metric space (Y, φ) where $\nu = O(1)$; for instance for point estimators (e.g., the geometric median), define a projection into $\mathbb{R}^1$ so no z_is map to the same point, then define distance φ as restricted to the distance along this line, so metric balls are intervals (or slabs in $\mathbb{R}^d$); these have $\nu = 2$.

4.1 Application to geometric median

For each $Q \Subset \mathcal{P}$, the geometric median m_Q may take a distinct value. Thus even calculating that set, let alone their weights in the case of duplicates, would require at least $\Omega(k^n)$ time. But it is straightforward to apply this randomized approach. For $P_{\mathsf{flat}} \in \mathbb{R}^d$, the natural metric space (Y, φ) is $Y = \mathbb{R}^d$ and φ as the Euclidian distance.

However, there is no known closed form solution for the geometric median; it can be computed within any additive error ϕ through various methods [22, 9, 7, 6]. As such, we can state a slightly more intricate corollary.

▶ **Corollary 13.** *Set $\varepsilon > 0$ and $\delta \in (0,1)$ and $N = O((1/\varepsilon^2)(d + \log(1/\delta)))$. For an uncertain point set $\mathcal{P}$ with $P_{\mathsf{flat}} \subset \mathbb{R}^d$, let the estimator E be the geometric median, and let E_ϕ be an algorithm that finds an approximation to the geometric median within additive error $\phi > 0$. Run the algorithm using E_ϕ. Then for any ball $B = B(x, r) \in \mathcal{R}$, there exists[2] another ball $B' = B(x, r')$ with $|r - r'| \leq \phi$ such that with probability at least $1 - \delta$, $\left| \sum_{z \in T \cap B'} w(z) - \Pr_{Q \in \mathcal{P}}[E(Q) \in B] \right| \leq \varepsilon$.*

4.2 Application to Siegel estimator

The Siegel (repeated median) estimator [19] is a robust estimator S for linear regression in $\mathbb{R}^2$ with optimal breakdown point 0.5. For a set of points Q, for each $q_i \in Q$ it computes slopes of all lines through q_i and each other $q' \in Q$, and takes their median a_i. Then it takes the median a of the set $\{a_i\}_i$ of all median slopes. The offset b of the estimated line $\ell : y = ax + b$, is the median of $(y_i - ax_i)$ for all points $q_i = (x_i, y_i)$. For uncertain data $P_{\mathsf{flat}} \subset \mathbb{R}^2$, we can directly apply our general technique for this estimator.

We use the following metric space (Y, φ). Let $Y = \{\ell \mid \ell \text{ is a line in } \mathbb{R}^2 \text{ with form } y = ax + b, \text{ where } a, b \in \mathbb{R}\}$. Then let φ be the Euclidean distance in the standard dual; for two lines $\ell : y = ax + b$ and $\ell' : y = a'x + b'$, define $\varphi(\ell, \ell') = \sqrt{(a - a')^2 + (b - b')^2}$. By examining the dual space, we see that $(Y, \mathcal{R})$ with $\mathcal{R} = \{B(\ell, r) \mid \ell \in Y, r \geq 0\}$ and $B(\ell, r) = \{\ell' \in Y \mid \varphi(\ell, \ell') \leq r\}$ has a VC-dimension 3.

From the definition of the Siegel estimator [19], there can be at most $O(n^3 k^3)$ distinct lines in $T = \{S(Q) \mid Q \Subset \mathcal{P}\}$. By Corollary 12, setting $N = O((1/\varepsilon^2)\log(1/\delta))$, then with probability at least $1 - \delta$ for all $z \in T$ we have $\left| w(z) - \Pr_{Q \in \mathcal{P}}[S(Q) = z] \right| \leq \varepsilon$.

[2] To simplify the discussion on degenerate behavior, define ball B', so any point q on its boundary can be defined inside or outside of B, and this decision can be different for each q, even if they are co-located.

References

1. Pankaj K. Agarwal, Boris Aronov, Sariel Har-Peled, Jeff M. Phillips, Ke Yi, and Wuzhou Zhang. Nearest-neighbor searching under uncertainty II. In *PODS*, 2013.
2. Pankaj K. Agarwal, Siu-Wing Cheng, Yufei Tao, and Ke Yi. Indexing uncertain data. In *PODS*, 2009.
3. Pankaj K. Agarwal, Alon Efrat, Swaminathan Sankararaman, and Wuzhou Zhang. Nearest-neighbor searching under uncertainty. In *PODS*, 2012.
4. Pankaj K. Agarwal, Sariel Har-Peled, Subhash Suri, Hakan Yildiz, and Wuzhou Zhang. Convex hulls under uncertainty. In *ESA*, 2014.
5. Greg Aloupis. Geometric measures of data depth. In *Data Depth: Robust Multivariate Analysis, Computational Geometry and Applications*. AMS, 2006.
6. Sanjeev Arora, Prabhakar Raghavan, and Satish Rao. Approximation schemes for Euclidean k-medians and related problems. In *STOC*, 1998.
7. Prosenjit Bose, Anil Maheshwari, and Pat Morin. Fast approximations for sums of distances clustering and the Fermet-Weber problem. *CGTA*, 24:135–146, 2003.
8. Kevin Buchin, Jeff M. Phillips, and Pingfan Tang. Approximating the distribution of the median and other robust estimators on uncertain data. *ArXiv e-prints*, 2018. `arXiv:1601.00630`.
9. R. Chandrasekaran and A. Tamir. Algebraic optimization: The Fermet-Weber location problem. *Mathematical Programming*, 46:219–224, 1990.
10. Graham Cormode and Andrew McGregor. Approximation algorithms for clustering uncertain data. In *PODS*, 2008.
11. David Donoho and Peter J. Huber. The notion of a breakdown point. In P. Bickel, K. Doksum, and J. Hodges, editors, *A Festschrift for Erich L. Lehmann*, pages 157–184. 1983.
12. Lingxiao Huang and Jian Li. Approximating the expected values for combinatorial optimization problems over stochastic points. In *ICALP*, 2015.
13. Allan G. Jørgensen, Maarten Löffler, and Jeff M. Phillips. Geometric computation on indecisive points. In *WADS*, 2011.
14. Jian Li, Barna Saha, and Amol Deshpande. A unified approach to ranking in probabilistic databases. In *VLDB*, 2009.
15. Yi Li, Philip M. Long, and Aravind Srinivasan. Improved bounds on the samples complexity of learning. *Journal of Computer and System Science*, 62:516–527, 2001.
16. Maarten Löffler and Jeff Phillips. Shape fitting on point sets with probability distributions. In *ESA*, 2009.
17. Hendrik P. Lopuhaa and Peter J. Rousseeuw. Breakdown points of affine equivaniant estimators of multivariate location and converiance matrices. *The Annals of Statistics*, 19:229–248, 1991.
18. Peter J. Rousseeuw. Multivariate estimation with high breakdown point. *Mathematical Statistics and Applications*, pages 283–297, 1985.
19. Andrew F. Siegel. Robust regression using repeated medians. *Biometrika*, 82:242–244, 1982.
20. J. W. Tukey. Mathematics and the picturing of data. In *Proceedings of the 1974 International Congress of Mathematics, Vancouver*, volume 2, pages 523–531, 1975.
21. Vladimir Vapnik and Alexey Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Th. Probability and Applications*, 16:264–280, 1971.
22. Endre Weiszfeld. Sur le point pour lequel la somme des distances de n points donnés est minimum. *Tohoku Mathematical Journal, First Series*, 43:355–386, 1937.
23. Ying Zhang, Xuemin Lin, Yufei Tao, and Wenjie Zhang. Uncertain location based range aggregates in a multi-dimensional space. In *Proceedings 25th IEEE International Conference on Data Engineering*, 2009.

Consistent Sets of Lines with no Colorful Incidence

Boris Bukh[1]
Carnegie Mellon University, Department of Mathematical Sciences
Pittsburgh, PA 15213, USA.
bbukh@math.cmu.edu

Xavier Goaoc[2]
Université Paris-Est, LIGM)
UMR 8049, CNRS, ENPC, ESIEE, UPEM, F-77454, Marne-la-Vallée, France.
xavier.goaoc@u-pem.fr

Alfredo Hubard
Université Paris-Est, LIGM
UMR 8049, CNRS, ENPC, ESIEE, UPEM, F-77454, Marne-la-Vallée, France.
alfredo.hubard@u-pem.fr

Matthew Trager[3]
PSL Research University
Inria, École Normale Supérieure, and CNRS.
matthew.trager@inria.fr

Abstract

We consider incidences among colored sets of lines in $\mathbb{R}^d$ and examine whether the existence of certain concurrences between lines of k colors force the existence of at least one concurrence between lines of $k+1$ colors. This question is relevant for problems in 3D reconstruction in computer vision.

2012 ACM Subject Classification Theory of computation → Randomness, geometry and discrete structures, Computing methodologies → Scene understanding

Keywords and phrases Incidence geometry, image consistency, probabilistic construction, algebraic construction, projective configuration

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.17

Related Version A full version of this paper is available at `https://arxiv.org/abs/1803.06267`.

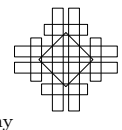
Funding Part of the work was done during the visit of BB to Université Paris-Est Marne-la-Vallée supported by LabEx Bézout (ANR-10-LABX-58).

Acknowledgements We thank Éric Colin de Verdière and Vojta Kalusza for discussions at an early stage of this work.

[1] Supported in part by Sloan Research Fellowship and by U.S. taxpayers through NSF CAREER grant DMS-1555149.
[2] Supported by Institut Universitaire de France.
[3] Supported in part by the ERC grant VideoWorld and the Institut Universitaire de France

34th International Symposium on Computational Geometry (SoCG 2018).
Editors: Bettina Speckmann and Csaba D. Tóth; Article No. 17; pp. 17:1–17:14

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



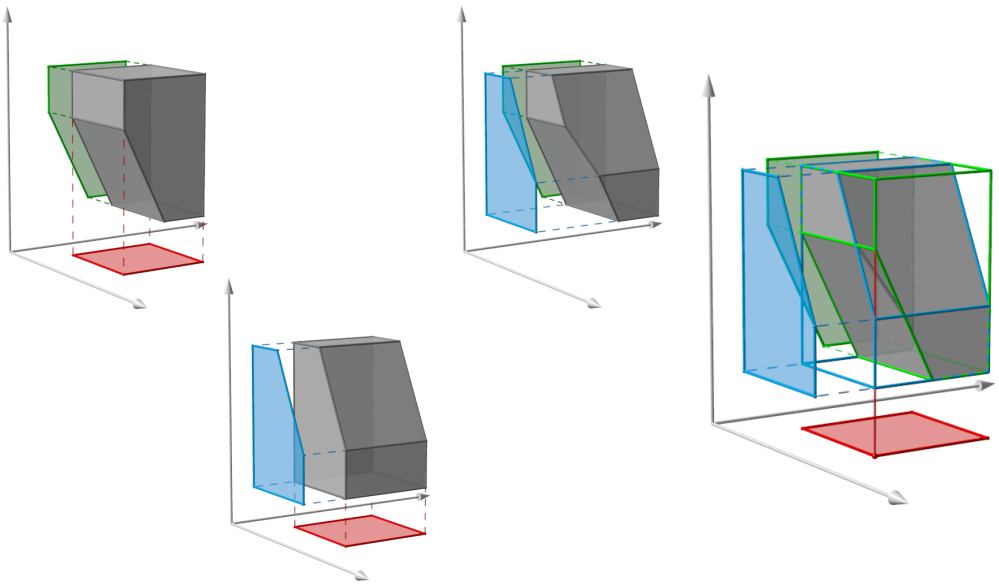


Figure 1 Three silhouettes that are 2-consistent but not globally consistent for three orthogonal projections. Each of the first three figures shows a three-dimensional set that projects onto two of the three silhouettes. The fourth figure illustrates that no set can project simultaneously onto all three silhouettes: the highlighted red image point cannot be lifted in 3D, since no point that projects onto it belongs to the pre-images of both the blue and green silhouettes.

1 Introduction

A central problem in computer vision is the reconstruction of a three-dimensional scene from multiple photographs. Trager et al. [16, Definition 1] defined a set of images as consistent if they represent the same scene from different points of view. They constructed examples (like that of Figure 1) of a set of images which is pairwise consistent while being altogether inconsistent. They also showed [16, Proposition 4] that under a certain convexity hypothesis, images that are consistent three at a time are globally consistent. In this paper we drop the convexity condition and consider these affairs from the point of view of incidence geometry.

Problem statement. An *incidence* is a set of lines that meet at a single point. Let $\mathcal{L} = \mathcal{L}_1 \cup \mathcal{L}_2 \cup \ldots \cup \mathcal{L}_m$ be a set of lines of m colors in $\mathbb{R}^d$ (where each $\mathcal{L}_i$ is a color class). Given $S \subset \{1, 2 \ldots m\}$, an *$S$-incidence* in $\mathcal{L}$ is an incidence between lines of every color in S. This paper focuses on the following notions:

▶ **Definition 1.** For $1 \leq k \leq m$, a *k-incidence* in $\mathcal{L}$ is a S-incidence where $|S| = k$. A *colorful incidence* in $\mathcal{L}$ is an incidence that contains lines of every color.

▶ **Definition 2.** The set $\mathcal{L}$ is *k-consistent* if for every k-tuple of colors $S \subset \{1, 2 \ldots m\}$, every line in $\cup_{i \in S} \mathcal{L}_i$ belongs to an S-incidence. The set $\mathcal{L}$ is *consistent* if every line belongs to (at least) one colorful incidence.

Instead of wondering if k-consistency implies consistency, we aim for a more modest goal:

▶ **Problem 3.** *Under which conditions does the k-consistency assumption imply the existence of a $(k+1)$-incidence?*

The main results of this paper are two constructions of (infinite families of) finite sets of lines which are k-consistent and have no colorful incidence. Thus, consistency does not propagate.

▶ Remark. Unless indicated otherwise, the set $\mathcal{L}$ is assumed to be finite. We also assume throughout that the lines in $\mathcal{L}$ are pairwise distinct. This has no consequence on Problem 3: repeating a line in a color class is useless, and if two lines of distinct colors coincide, then the k-consistency assumption trivially implies that this line has a $(k+1)$-incidence.

Relation to photograph consistency. Let us explain how our initial image consistency question relates to Problem 3. Firstly, we ignore color or intensity information, and treat the scene as a set of opaque objects and the images as their projections onto certain planes. In this setting, images are *consistent* if and only if there exists a subset $R \subset \mathbb{R}^3$ that projects into each of them. Assuming that light travels along straight lines, the set of 3D points that are mapped to a given image point is a ray, or more conveniently a line, in $\mathbb{R}^3$. Starting with m photographs, if we let $\mathcal{L}_i$ denote the lines that are pre-images of the projection on the ith photograph, then the photographs are consistent if and only if $\cup_{i=1}^{m} \mathcal{L}_i$ is consistent: R is the set of points of colorful incidences.

Setting. In the basic set-up for computer vision, all lines used to project the scene onto a given image plane pass through a "pinhole". We therefore define a color class $\mathcal{L}_i$ as *concurrent* if it consists of concurrent lines. We consider, however, the problem more generally since it is possible to build other imaging systems. For example, there are cameras that use the lines secant to two fixed skew lines; other cameras use the lines secant to an algebraic curve γ and to a line intersecting γ in $\deg \gamma - 1$ points. For a discussion of the geometry of families of lines arising in the modeling of imaging systems, see [2, 17] and the references therein.

We focus in this paper on the consistency of finite sets of lines. This restriction is technically convenient and remains relevant to the initial motivation on continuous sets of lines. On the one hand, our constructions for the finite problem turn out to readily extend to infinite families of lines (see Section 4). On the other hand, the finite problem is already relevant to 3D reconstruction, when one has to recover the camera parameters (settings or position) used in the photographs. Indeed, this recovery is typically done by identifying pixels in different images that are likely to be the projection of the same 3D element, and using the incidence structure of their inverse images to infer the position of the camera; this process is called *structure from motion* [14]. The number of lines required to determine the cameras is typically 5 to 7 per image. Although pixels are usually matched across pairs of images, there are good reasons for wanting to match them across more images, firstly for robustness to noise, but also because this avoids ambiguities in the reconstruction in the case of degenerate camera configurations (for example, pairwise matches are never sufficient to reconstruct a scene from images when all the camera pinholes are exactly aligned [12, Chapter 15.4.2]). Understanding the consistency propagation may simplify the certification of such matchings.

1.1 Results

We focus on Problem 3 for $k \geq 3$ because examples of tricolor sets of lines that are 2-consistent but without a colorful incidence are relatively easy to build:

▶ **Example 4.** Let $(\vec{x}_0, \vec{x}_1, \vec{x}_2)$ be a basis of $\mathbb{R}^3$. Let $p_0, p_2 \ldots, p_{3n}$ be a set of points where p_0 is arbitrary, $p_{i+1} \in p_i + \mathbb{R}\vec{x}_{(i \bmod 3)}$ and $p_{3n} = p_0$. For each $i \in \{0, 1, 2\}$ define $\mathcal{L}_i$ to be the set of lines in coordinate direction i that are incident to points p_j with $j \equiv i \pmod 3$ and $j \equiv i - 1 \pmod 3$. If desired, we may apply a projective transformation that turns parallelism into concurrence.

Constructions from higher-dimensional grids. We present two constructions of arbitrary large sets of lines in $\mathbb{R}^d$ of $k+1$ colors that are k-consistent and have no colorful incidence, for every $k \geq 3$ and $k+1 \geq d \geq 2$. Both constructions are based on selecting subsets of lines from a regular grid in $\mathbb{R}^{k+1}$. In one case, the selection is probabilistic (Theorems 5), while in the other case it uses linear algebra over finite vector spaces (Theorem 6). In both constructions, every color class is concurrent. The probabilistic argument is asymptotic and proves the existence of configurations where every line is involved in many k-incidences for every choice of $k-1$ other colors. The algebraic construction is explicit and is minimal in the sense that removing any line breaks the k-consistency.

Restrictions on higher-dimensional grids. We then test the sharpness and potential of constructions from higher-dimensional grids. On the one hand, we examine the number of lines of such constructions. The algebraic selection method picks at least 2^{k^2-k-1} lines of each color (we leave aside the probabilistic selection method as its analysis is asymptotic). This construction has the property that the lines meeting at a k-incidence are not "flat", in the sense that they are not contained in a $k-1$-dimensional subspace. We show, using the polynomial method [11], that for any construction with this property, the number of lines must be at least exponential in k (Proposition 7). On the other hand, we examine the possibility of designing similar constructions for models of cameras in which the lines are not all concurrent. We observe that when every color class is secant to two fixed lines, lines from two color classes cannot form a complete bipartite intersection graph (Proposition 8).

Small configurations. We also investigate small-size configurations of lines in $\mathbb{R}^3$ with 4 colors that are 3-consistent but have no colorful incidence. The smallest example provided by our constructions has 32 lines per color, which says little for applications like structure from motion, where each color class has very few lines. Figure 2 shows two non-planar examples with 12 lines each. We prove that they are the only non-planar constructions with these parameters (Theorem 11). We also show that any configuration with these parameters and concurrent color classes must have at least 24 lines or be planar (Theorem 10).

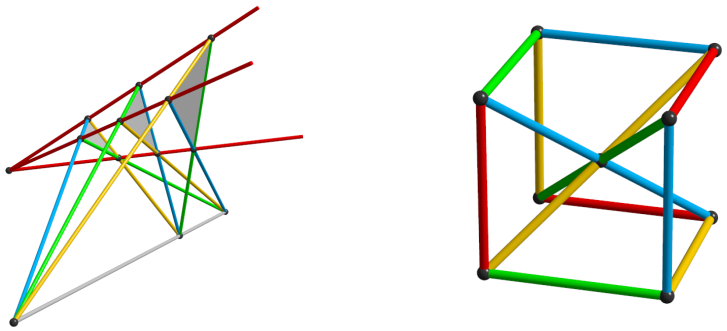


Figure 2 Two non-planar examples of 12 lines in 4 colors that are 3-consistent and have no 4-incidence. (Left) A variation around Desargues' configuration. (Right) A subset of the $(12_4 16_3)$ configuration of Reye; note that triples of parallel lines intersect at infinity.

1.2 Related work

The study of consistent families of colored lines relates most prominently to classical questions in computer vision and in discrete geometry.

In computer vision. The simplest and most extensively studied setting for consistency deals with families where each color class has a single line. The study of n-tuples of lines that are incident at a point (or "point correspondences"), is central in *multi-view geometry* [12], that is the foundation of 3D-reconstructions algorithms. In this setting, consistency propagates trivially: n lines are concurrent if and only if any three of them are (even better: n lines not all coplanar are concurrent if and only if every pair of them is). Concurrency constraints are traditionally expressed algebraically as polynomials in image coordinates (see, *e.g.* [6]).

A more systematic study of consistency for silhouettes (*i.e.*, for infinite families of lines) was proposed to design reconstruction methods based on shapes more complex than points or lines [3, 13]. Pairwise consistency for silhouettes can be encoded in a "generalized epipolar constraint", which can be viewed as an extension of the epipolar constraint for points, and expresses 2-consistency in terms of certain simple tangency conditions [1, 16]. There is no known similar characterization for k-consistency with $k > 2$. Consistency propagation is only known for convex silhouettes: 3-consistency implies consistency [16].

In the dual, consistency expresses conditions for a family of planar sets to be sections of the same 3D object [16], a question classical in geometric tomography or stereology. We are not aware of any relevant result on consistency in these directions.

Discrete geometry. As evidenced by Figure 2, our analysis of small configurations relates to the classical configurations of Reye and Desargues in projective geometry. Our problem and results for larger configurations relate to various lines of research in incidence geometry. Inspired by the Sylvester–Gallai theorem, Erdős [5] asked for the largest number of collinear k-tuples in a planar point set with no collinear $k+1$-tuple. The best construction for $k = 3$ come from irreducible cubic curves[4]. For higher k the best constructions were given by Solymosi and Stojakovíc [15] and are projections of higher-dimensional subsets of the regular grid (selected, unlike ours, by taking concentric spheres). In the plane, our problem is dual to a colorful variant of Erdős's question. An intermediate between Erdős's problem and the one treated here would ask for the existence of a set of lines $\mathcal{L}$ in which each line is involved in many (colorless) k-incidences but there are no (colorless) $k+1$-incidences. Since the Solymosi-Stojakovíc construction provides $n^{2-\frac{c}{\sqrt{n}}}$ aligned k tuples of points, it is not hard to see, using a greedy deletion argument, that this alternative problem is essentially equivalent to Erdős's original one.

In higher dimensions, the question of finding sets of lines with many k-rich points (in the terminology of [10]) is interesting even without the condition of having no $(k+1)$rich point. Much of the recent research around this question has followed the solution to the joint problem [11] and has been driven by algebraic considerations (see [10] and the references therein). Here, we also ask for many k-rich points, but our questions are driven by combinatorial considerations. Our assumptions trade the usual density requirements (we assume linearly many, rather than polynomially many, k-rich points) for structural hypotheses in the form of conditions on the colors. We can still use some of the algebraic methods; the proof of Proposition 7 is, for instance, modeled on the upper bound on the number of joints of Guth and Katz [11].

[4] This case is closely connected with the famous *orchard problem* recently solved in its asymptotic version [8]

2 Probabilistic construction

In this section we prove:

▶ **Theorem 5.** *For any $k \geq 3$, $k+1 \geq d \geq 2$, and arbitrarily large $N \in \mathbb{N}$, there exists a finite set of lines in $\mathbb{R}^d$ of $k+1$ colors that is k-consistent, has no $(k+1)$-incidence, and in which each color class consists of between N and $3N$ lines, all concurrent.*

We describe our construction in $\mathbb{R}^{k+1}$ with color classes consisting of parallel lines. We then apply an adequate projective transform (to turn parallelism into concurrence) and a generic projection to a d-dimensional space; both transformations preserve incidences and therefore the properties of the construction.

Construction. Consider the finite subset $[n]^{k+1} = \{1, 2, ..., n\}^{k+1} \subset \mathbb{R}^{k+1}$ of the integer grid. We make our construction in two stages:

- Consider the set $\mathcal{L}_i^{\#}$ of n^k lines that are parallel to the ith coordinate axis and contain at least one point of our grid. We pick a random subset $\mathcal{L}'_i$, where each line from $\mathcal{L}_i^{\#}$ is chosen to be in $\mathcal{L}'_i$ independently with probability $p \stackrel{\text{def}}{=} 2n^{-\frac{2}{2k-1}}$ (the value of p is chosen with foresight).
- We then delete from $\mathcal{L}'_i$ all lines that are concurrent with k other lines from $\cup_{j\neq i}\mathcal{L}'_j$ and denote the resulting set $\mathcal{L}_i$.

We let $\mathcal{L}$ denote the colored set of lines $\mathcal{L} = \mathcal{L}_1 \cup \mathcal{L}_2 \cup \ldots \cup \mathcal{L}_{k+1}$. The second stage of the construction ensures that $\mathcal{L}$ has no $(k+1)$-incidence.[5] To prove Theorem 5, it thus suffices to show that with positive probability, $\mathcal{L}$ is k-consistent and each $\mathcal{L}_i$ has the announced size. Let us clarify that all lines considered in the proof are in $\cup_{i=1}^{k+1}\mathcal{L}_i^{\#}$ unless stated otherwise.

Consistency. Let us argue that $\mathcal{L}$ is k-consistent with high probability. For a set $I \subset [k+1]$, let

$$S_I \stackrel{\text{def}}{=} \{Q \in [n]^{k+1} : \forall i \in I \text{ there is a line of } \mathcal{L}_i \text{ containing } Q\},$$
$$S'_I \stackrel{\text{def}}{=} \{Q \in [n]^{k+1} : \forall i \in I \text{ there is a line of } \mathcal{L}'_i \text{ containing } Q\}.$$

We say that $\ell \in \mathcal{L}_i^{\#}$ is *j-bad* (for $j \neq i$) if ℓ contains no point of $S_{[k+1]\setminus\{i,j\}}$. Note that $\mathcal{L}$ is not k-consistent precisely when some $\ell \in \mathcal{L}_i^{\#}$ is j-bad and ℓ ends up in $\mathcal{L}_i$.

Let $\ell \in \mathcal{L}_i^{\#}$ and let $L \subset \mathcal{L}_i^{\#}$ be any set containing ℓ. Let $j \neq i$. We shall estimate $\mathrm{P}\left[(\ell \in \mathcal{L}_i) \wedge (\ell \text{ is } j\text{-bad}) \mid \mathcal{L}'_i = L\right]$. For ease of notation, we may assume that $i = k+1$, $j = k$ and ℓ is the line $\{(1, 1, \ldots, 1, x) : x \in \mathbb{R}\}$. Call a point $Q \in [n]^{k+1}$ *regular* if $Q \notin \ell$.

The randomness in the construction comes from $(k+1)n^k$ random choices, one for each line in $\mathcal{L}_1^{\#} \cup \cdots \cup \mathcal{L}_{k+1}^{\#}$. We refer to these random choices as 'coin flips' since we can think of each as being a result of a toss of a (biased) coin.

Let $\ell_{r,x}$ denote the line $\{(1, 1, \ldots, 1, y, 1, \ldots, 1, x) : y \in \mathbb{R}\}$, where y is at position r. If a line $\ell' \notin \mathcal{L}_r^{\#}$ intersects $\ell_{r,x}$ in point $(1, 1, \ldots, 1, y, 1, \ldots, 1, x)$, then all points of ℓ' have y in the rth position. Note that a point $(1, 1, \ldots, 1, y, 1, \ldots, 1, x)$ is regular if $y \neq 1$. A crucial observation is that if a line $\ell' \notin \mathcal{L}_{k+1}^{\#}$ intersects $\ell_{r,x}$ in a regular point and a line $\ell'' \notin \mathcal{L}_{k+1}^{\#}$ intersects $\ell_{r',x'}$ in a regular point and $(r, x) \neq (r', x')$, then ℓ' is different from ℓ''. This implies that sets of coin flips on which the events of the form

[5] Deleting one line per concurrence of size $k+1$ would suffice, but deleting all lines as we do simplifies the analysis and suffices for our purpose.

$$\text{“there is a regular } Q \in \ell_{r,x}\ Q \in S'_{[k]\setminus\{r\}}\text{”}$$

are disjoint for distinct (r,x), apart from the flips associated to the lines in $\mathcal{L}^{\#}_{k+1}$.

For a point $Q \in [n]^{k+1}$, let $\lambda(Q)$ be the line in $\mathcal{L}^{\#}_{k+1}$ containing Q. Hence,

$$\begin{aligned}
&\mathrm{P}\left[(\ell \in \mathcal{L}'_{k+1}) \wedge (\ell \text{ is } k\text{-bad}) \mid \mathcal{L}'_{k+1} = L\right] \\
&= \mathrm{P}\left[(\ell \in \mathcal{L}'_{k+1}) \wedge \bigwedge_{x\in[n]} (1,1,\dots,1,x) \notin S_{[k-1]} \mid \mathcal{L}'_{k+1} = L\right] \\
&= \mathrm{P}\left[(\ell \in \mathcal{L}'_{k+1}) \wedge \bigwedge_{x\in[n]} (\exists r \in [k-1]\ \ell_{r,x} \notin \mathcal{L}_r) \mid \mathcal{L}'_{k+1} = L\right] \\
&= \mathrm{P}\left[(\ell \in \mathcal{L}'_{k+1}) \wedge \bigwedge_{x\in[n]} \left(\exists r \in [k-1]\ \ell_{r,x} \notin \mathcal{L}'_r \vee (\exists Q \in \ell_{r,x} \cap S'_{[k+1]})\right) \mid \mathcal{L}'_{k+1} = L\right]
\end{aligned}$$

In this last formula, the point Q can be assumed to be regular because $\ell \in L$, by assumption. Now we may drop $\ell \in \mathcal{L}'_{k+1}$ to obtain that the above is

$$\leq \mathrm{P}\left[\bigwedge_{x\in[n]} \left(\exists r \in [k-1]\ \ell_{r,x} \notin \mathcal{L}'_r \vee (\exists \text{reg. } Q \in \ell_{r,x} \cap S'_{[k+1]})\right) \mid \mathcal{L}'_{k+1} = L\right]$$

Observe that if $\ell_{r,x} \in \mathcal{L}'_r$ then $Q \in \ell_{r,x} \cap S'_{[k+1]}$ holds if and only if $Q \in \ell_{r,x} \cap S'_{[k]\setminus\{r\}}$ and $\lambda(Q) \in L$. By the observation above, the set of coin flips on which these latter events depend for different x are disjoint, so this probability is

$$\begin{aligned}
&= \prod_{x\in[n]} \mathrm{P}\left[\exists r \in [k-1]\ \ell_{r,x} \notin \mathcal{L}'_r \vee (\exists \text{reg. } Q \in \ell_{r,x} \cap S'_{[k]\setminus\{r\}} \wedge \lambda(Q) \in L) \mid \mathcal{L}'_{k+1} = L\right] \\
&= \prod_{x\in[n]} \Bigg(1 - \mathrm{P}\Big[\forall r \in [k-1]\ \ell_{r,x} \in \mathcal{L}'_r \\
&\qquad\qquad \wedge (\forall \text{reg. } Q \in \ell_{r,x} Q \notin S'_{[k]\setminus\{r\}} \vee \lambda(Q) \notin L) \mid \mathcal{L}'_{k+1} = L\Big]\Bigg) \\
&= \prod_{x\in[n]} \left(1 - \prod_{r\in[k-1]} \left(p \cdot \prod_{\substack{\text{regular } Q\in\ell_{r,x} \\ \lambda(Q)\in L}} \mathrm{P}\left[Q \notin S'_{[k]\setminus\{r\}}\right]\right)\right)
\end{aligned}$$

Call $L \subset \mathcal{L}^{\#}_{k+1}$ *unbiased* if for every pair $(r,x) \in [k-1]\times[n]$ the number of points $Q \in \ell_{r,x}$ such that $\lambda(Q) \in L$ is at most $2pn$. For unbiased L, we obtain that the above is at most

$$\left(1 - \left(p \cdot (1-p^{k-1})^{2pn}\right)^{k-1}\right)^n \leq \left(1 - \left(\tfrac{1}{2}p\right)^{k-1}\right)^n \leq e^{-n\left(\frac{1}{2}p\right)^{k-1}} = e^{-n^{\frac{1}{2^{k-1}}}}$$

If we pick L uniformly at random, then, for every $(r,x) \in [k-1]\times[n]$, the number of points $Q \in \ell_{r,x}$ such that $\lambda(Q) \in L$ is a binomial random variable. Chernoff's bound then yields

$$\begin{aligned}
&\mathrm{P}\left[(\ell \in \mathcal{L}'_{k+1}) \wedge (\ell \text{ is } k\text{-bad})\right] \\
&\leq \mathrm{P}\left[\mathcal{L}_{k+1} \text{ is biased}\right] + \sum_{\text{unbiased } L} \mathrm{P}\left[\mathcal{L}'_{k+1} = L\right] \mathrm{P}\left[(\ell \in \mathcal{L}'_{k+1}) \wedge (\ell \text{ is } k\text{-bad}) \mid \mathcal{L}'_{k+1} = L\right] \\
&\leq \sum_{(r,x)\in[k-1]\times[n]} e^{-(pn)^2/2n} + e^{-n^{\frac{1}{2k-1}}} = e^{-cn^{\frac{1}{2k-1}}}.
\end{aligned}$$

By taking the union bound over all i, j and ℓ we obtain that

$$\begin{aligned}
\mathrm{P}\left[\mathcal{L} \text{ is not } k\text{-consistent}\right] &\leq \mathrm{P}\left[\exists i, j\ \exists \ell \in \mathcal{L}_i^{\#}\ \left((\ell \in \mathcal{L}'_i) \wedge (\ell \text{ is } j\text{-bad})\right)\right] \\
&\leq (k+1)^2 n^k e^{-cn^{\frac{1}{2k-1}}} \leq e^{-c' n^{\frac{1}{2k-1}}}.
\end{aligned}$$

Size. We now analyze the probability that $\mathcal{L}_1$ is large (the bound will hold for each $\mathcal{L}_i$). Let us write $\mathcal{L}' = \cup_{i=1}^{k+1} \mathcal{L}'_i$ and label $\ell_1, \ell_2, \ldots, \ell_{n^k}$ the lines parallel to the 1st coordinate axis that intersect our grid. Put $X_i = \mathbb{1}_{\ell_i \in \mathcal{L}_1}$ and let $X = |\mathcal{L}_1| = X_1 + X_2 + \ldots + X_{n^k}$. We have

$$\mathrm{E}\left[X_i\right] = \mathrm{P}\left[X_i = 1\right] = \mathrm{P}\left[\ell_i \in \mathcal{L}'\right] \mathrm{P}\left[\ell_i \in \mathcal{L} \mid \ell_i \in \mathcal{L}'\right] = p(1-p^k)^n, \text{ and}$$

$$\mathrm{E}\left[X\right] = n^k p(1-p^k)^n = \left(1 - n^{-\frac{2k}{2k-1}}\right)^n n^{k-\frac{2}{2k-1}} \geq \left(1 - \frac{1}{n}\right)^n n^{k-\frac{2}{2k-1}} \geq \frac{1}{4} n^{k-\frac{2}{2k-1}}.$$

Thus $\mathrm{E}\left[X\right] = N \in [\frac{1}{4} n^{k-\frac{2}{2k-1}}, n^{k-\frac{2}{2k-1}}]$. We next use a concentration inequality to pass from $\mathrm{E}\left[X\right]$ to an estimate on the probability that X is large. The second step introduces some dependency between some of the variables X_i, so we use Chebychev's inequality:

$$\mathrm{P}\left[|X - \mathrm{E}\left[X\right]| > \frac{1}{2}\mathrm{E}\left[X\right]\right] \leq 4\frac{\mathrm{Var}\left[X\right]}{\mathrm{E}\left[X\right]^2} \leq 64 \mathrm{Var}\left[X\right] n^{-\left(2k-\frac{4}{2k-1}\right)}.$$

Recall that

$$\mathrm{Var}\left[X\right] = \sum_{i=1}^{n^k} \mathrm{Var}\left[X_i\right] + \sum_{1\leq i<j\leq n^k} \mathrm{Cov}\left[X_i, X_j\right].$$

Since X_i takes values in $\{0,1\}$, the first sum in the right-hand term is bounded by n^k. Moreover, there are $O(n^{k+1})$ pairs of variables X_i and X_j with non-zero covariance, since this requires the two lines ℓ_i and ℓ_j to belong to a common axis-aligned 2-plane. Again, each non-zero covariance is at most 1. Altogether, $\mathrm{Var}\left[X\right] = O(n^{k+1})$, so

$$\mathrm{P}\left[|X - \mathrm{E}\left[X\right]| > \frac{1}{2}\mathrm{E}\left[X\right]\right] = O\left(n^{\frac{2k+3}{2k-1}-k}\right).$$

For $k \geq 3$, the probability that X is in $\left[\frac{1}{8} n^{k-\frac{2}{2k-1}}, \frac{3}{2} n^{k-\frac{2}{2k-1}}\right]$ goes to 1 as n goes to infinity.

3 Algebraic construction

In this section we prove:

▶ **Theorem 6.** *For any $k \geq 3$, $k+1 \geq d \geq 2$, and arbitrarily large N, there exists a finite set of lines in $\mathbb{R}^d$ with $k+1$ colors that is k-consistent, has no $(k+1)$-incidence, and in which each color class consists of N lines, all concurrent.*

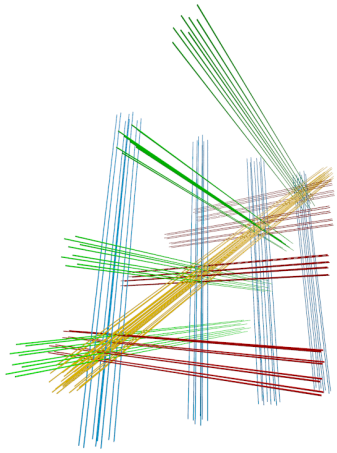


Figure 3 A projection to $\mathbb{R}^3$ of our construction for $k = 3$ and $p = 2$ (reprojected to the plane).

As in Section 2, we describe our construction in $\mathbb{R}^{k+1}$ with parallel families of lines, and obtain the desired configuration by an adequate projective transformation and a projection. We again consider the finite portion of the integer grid $[n]^{k+1} \subset \mathbb{R}^{k+1}$ and the axis-aligned lines that intersects it. Unlike in Section 2, we give an explicit way to select some of these lines to achieve the desired configuration.

Construction. We work with axis-aligned lines that intersect in points of our grid. Hence, identifying each line with the subset of the grid that it contains does not affect incidences. We fix a prime number p and parameterize $[n]$ by the vector space $\mathbb{V} = (\mathbb{Z}/p\mathbb{Z})^{k-1}$; this restricts the choice of n to certain prime powers, but still allows to make it arbitrarily large. We use this parameterization to describe the lines in our configuration as solutions of well-chosen linear equations.

Let $v_1, v_2, \ldots, v_k \in \mathbb{V}$ such that $v_1 + v_2 + \ldots + v_k = 0$ and any proper subset of them are linearly independent. Let $\cdot$ denote the inner product of the vector space $\mathbb{V}$. For $i = 1 \ldots k$, our set $\mathcal{L}_i$ consist of all the lines parallel to the ith coordinates and passing through a point with parameters $(X_1, \ldots, X_{k+1}) \in \mathbb{V}^{k+1}$ such that

$$v_{i-1} \cdot X_1 + v_{i-1} \cdot X_2 + \ldots + v_{i-1} \cdot X_{i-1} + v_i \cdot X_{i+1} + \ldots + v_i \cdot X_{k+1} = 0. \tag{1}$$

(Keep in mind that each X_i is a vector in $(\mathbb{Z}/p\mathbb{Z})^{k-1}$.) We define $\mathcal{L}_{k+1}$ similarly but replace Equation (1) by

$$v_k \cdot X_1 + v_k \cdot X_2 + \ldots + v_k \cdot X_k = 1. \tag{2}$$

No $(k+1)$-incidence. Any $(k+1)$-incidence is a point of the grid whose parameters $(X_1, \ldots, X_{k+1})$ satisfy the system:

$$\begin{cases}
 & v_1 \cdot X_2 & + & v_1 \cdot X_3 & + & \ldots & + & v_1 \cdot X_k & + & v_1 \cdot X_{k+1} & = 0 \\
v_1 \cdot X_1 & & + & v_2 \cdot X_3 & + & \ldots & + & v_2 \cdot X_k & + & v_2 \cdot X_{k+1} & = 0 \\
v_2 \cdot X_1 & + & v_2 \cdot X_2 & & + & \ldots & + & v_3 \cdot X_k & + & v_3 \cdot X_{k+1} & = 0 \\
v_3 \cdot X_1 & + & v_3 \cdot X_2 & + & v_3 \cdot X_3 & + & \ldots & + & v_4 \cdot X_k & + & v_4 \cdot X_{k+1} & = 0 \\
\ldots & & & & & & & & & & & \\
v_{k-1} \cdot X_1 & + & v_{k-1} \cdot X_2 & + & v_{k-1} \cdot X_3 & + & \ldots & & & + & v_k \cdot X_{k+1} & = 0 \\
v_k \cdot X_1 & + & v_k \cdot X_2 & + & v_k \cdot X_3 & + & \ldots & + & v_k \cdot X_k & & & = 1
\end{cases}$$

Summing all these conditions yields

$$\left(\sum_{i=1}^{k} v_i\right) \cdot \left(\sum_{i=1}^{k+1} X_i\right) = 1,$$

which contradicts $v_1 + v_2 + \ldots + v_k = 0$. So there is no $(k+1)$-incidence.

***k*-consistency.** Fix a line $\ell \in \mathcal{L}_1$. It corresponds to some solution $(X_2^*, \ldots, X_{k+1}^*)$ of Equation (1). The grid points on ℓ are precisely the points of the form $(X_1, X_2^*, X_3^*, \ldots, X_{k+1}^*)$ and are parameterized by X_1. Each equation in the system above reduces to $v_j \cdot X_1 = c_j$, where c_j is some constant vector (computed from the v_j's and the X_j^*'s). Since $X_1 \in (\mathbb{Z}/p\mathbb{Z})^{k-1}$ and any $k-1$ of the v_j are linearly independent, any choice of $k-1$ equations has a solution. This means that for any i, the line ℓ is concurrent with lines from all $\mathcal{L}_j$ with $j \in [k+1] \setminus \{i\}$. The same goes with the lines of $\mathcal{L}_2, \ldots, \mathcal{L}_{k+1}$ so the configuration is consistent.

Size. In this construction, the size of $\mathcal{L}_i$ is the number of $(X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_{k+1})$ in $\mathbb{V}^k$ satisfying Equation (1) – or (2) if $i = k+1$. Hence $|\mathcal{L}_i| = p^{k^2-k-1}$ for every i. The smallest configuration built in this way thus has 2^{k^2-k-1} lines per set (which is 32 for $k = 3$); refer to Figure 3.

4 More on grid-like examples

Both Theorems 5 and 6 construct examples as projections of subsets of a regular grid in higher dimension. We discuss here the properties of such constructions.

Number of lines. Consider a colored set of lines $\mathcal{L} = \mathcal{L}_1 \cup \mathcal{L}_2 \cup \ldots \cup \mathcal{L}_{k+1}$ in $\mathbb{R}^d$. We say that a t-incidence of $\mathcal{L}$ is *flat* if the lines meeting there are contained in an affine subspace of dimension at most $\min(d, t) - 1$. In any grid-like construction such as those in Theorems 5 and 6, every k-incidence is non-flat.

▶ **Proposition 7.** *Let $\mathcal{L} = \mathcal{L}_1 \cup \mathcal{L}_2 \cup \ldots \cup \mathcal{L}_m$ be a k-consistent colored set of lines in $\mathbb{R}^k$ with no $(k+1)$-incidence. If no k-incidence of $\mathcal{L}$ is flat, then*

$$\sum_{i=1}^{m} |\mathcal{L}_i| \geq \frac{\binom{\binom{m-1}{k-1}+k-1}{k}}{\binom{m-1}{k-1}}.$$

The proof essentially follows the argument of Guth and Katz [11] for bounding the number of joint among n lines; the main difference is that the consistency assumption makes their initial pruning step unnecessary. We spell out the details in [4, Proposition 7]. For $m = k+1$, the bound of Proposition 7 is $\frac{1}{k}\binom{2k}{k}$, so the number of lines required grows exponentially with k.

Non-concurrent colors. Theorems 5 and 6 both use a grid in $\mathbb{R}^{k+1}$ to start with $k+1$ color classes, each of size n^k, where every line is involved in n colorful incidences. Recall that in this setup, every color class is concurrent (it consists of parallel lines). This is in fact important, perhaps essential. To see this, note that any two of our starting color classes contain arbitrarily large subsets whose intersection graph is dense. This is impossible, generically, if we try to work with families of lines that are secant to two skew lines in $\mathbb{R}^3$. [6]

[6] This choice is motivated by the design of *two-slit cameras* [17, 2].

▶ **Proposition 8.** *For $i = 1, 2$, let Γ_i denote the set of lines secant to two fixed lines s_i and s'_i in $\mathbb{R}^3$. Let A and B be two sets of n lines from Γ_1 and Γ_2, respectively. If the lines s_1, s'_1, s_2 and s'_2 are in generic position then the intersection graph of A and B has $O\left(n^{4/3}\right)$ edges.*

Proof. First, note that the intersection graph of A and B is semi-algebraic: parameterizing Γ_i by $s_i \times s'_i \simeq \mathbb{R}^2$ makes the incidence an algebraic relation, as can be deduced from the bilinearity of incidence in Plücker coordinates. Next, remark that if this graph contains a complete bipartite subgraph $K_{3,3}$, then the lines $\{s_1, s'_1, s_2, s'_2\}$ are in a special position. Indeed, in the generic case, these two triples of lines come from the two families of rulings of a quadric surface [18, §10]; the lines s_1, s'_1, s_2 and s'_2 are also rulings of that quadric, so *both* s_1 and s'_1 intersect *both* s_2 and s'_2. In the non-generic cases, the six lines must be coplanar with s_1 and s_2. Now, we apply the semi-algebraic version of the Kővári –Sós–Turán theorem [7], and obtain that the number of edges of our graph is $O\left(n^{4/3}\right)$. ◀

We see the previous result as an indication that a straightforward adaptation of our probabilistic construction to the case of two-slits is unlikely. Can the bound in Proposition 8 be improved from $O\left(n^{4/3}\right)$ to $O(n)$?

▶ Remark. Note that the genericity assumption in Proposition 8 is on the sets Γ_i, not on their subsets. The analogue for concurrent sets of lines would be to require that the centers of concurrence are in generic position; this clearly does not prevent finding arbitrarily large subsets with dense intersection graphs.

Extension to continuous sets of lines. The constructions of Theorems 5 and 6 can be turned into continuous families of lines as follows.

First, we follow either construction up to the point where we have a family $\mathcal{L}$ of lines of $k+1$ colors in $\mathbb{R}^{k+1}$ that is k-consistent, without colorful incidence, and where each color class is parallel. Consider a parameter $\epsilon > 0$, to be fixed later. For every i, we build a set $\mathcal{L}_i(\epsilon)$ by considering every line $\ell \in \mathcal{L}_i$ in turn, and adding to $\mathcal{L}_i$ every line ℓ' parallel to ℓ such that the distance between ℓ and ℓ' is most ϵ. Note that for $\epsilon < 1/2$ the family $\mathcal{L}(\epsilon)$ is k-consistent and without colorful incidence.

Now, consider a generic projection $f\colon \mathbb{R}^{k+1} \to \mathbb{R}^d$ for the desired d. For any $\epsilon > 0$ the family $\mathcal{L}(\epsilon)$ is k-consistent. We observe that for $\epsilon > 0$ small enough, it also remains without colorful incidence. Let τ denote the minimum distance, in the projection, between a k-incidence and a line (of any color) not involved in that incidence. Every k-incidence in $\mathcal{L}$ gives rise, in $\mathcal{L}(\epsilon)$, to k tubes that intersect in a bounded convex set B of size $O(\epsilon)$. Choosing $\epsilon > 0$ such that the diameter of $f(B)$ is less than $\tau/2$ ensures that the corresponding family $f(\mathcal{L}(\epsilon))$ has no colorful incidence.

For a given family of colored lines $\mathcal{L}$ define the set P_S to be the set of points incident to at least one line of each of the color classes in S; see Figure 1. Notice that in our examples, for each set S of k colors the set P_S is highly disconnected. As mentioned in the introduction, Trager et al. [16] showed that if a family of sets of lines is 3-consistent and for each S of size 3, the set P_S is convex, then the whole family is consistent. An interesting open question is whether an analogue theorem holds if instead of convexity, we assume that for every set S of size k, the set P_S is sufficiently connected.

5 Constructions with few lines

The configurations constructed in Sections 2 and 3 have at least 32 lines per color. This is considerably larger than the sets of lines involved in some of the questions around consistency

that arise in computer vision. In the example of structure-from-motion mentioned in introduction, when the camera is central, every color class has only 5 to 7 lines. It turns out that for sufficiently small configurations, k-consistency does imply some colorful incidences:

▶ **Lemma 9.** *Any 3-consistent colored set of lines $\mathcal{L} = \mathcal{L}_1 \cup \mathcal{L}_2 \cup \mathcal{L}_3 \cup \mathcal{L}_4$ in $\mathbb{R}^d$ with $|\mathcal{L}_1| = |\mathcal{L}_2| = |\mathcal{L}_3| = |\mathcal{L}_4| = 2$ contains a colorful incidence.*

Proof. Let us prove the case where $d = 2$; the general case follows by projecting onto a generic 2-plane. Let P_i denote the dual of $\mathcal{L}_i$, and let $P = P_1 \cup P_2 \cup P_3 \cup P_4$. Assume, by contradiction, that $\mathcal{L}$ contains no colorful incidence, *i.e.* that no line intersects every P_i. Let $P' = P_2 \cup P_3 \cup P_4$ and let us apply a projective transform to map the points of P_1 to the horizontal and vertical directions, respectively. We call a line that contains a point of each of P_2, P_3 and P_4 a *rainbow line*.

Since $\mathcal{L}$ is 3-consistent, for any point $x \in P$ and any choice of 2 other colors, there is a line through x that contains a point of each of these colors. Since $\mathcal{L}$ has no colorful incidence, there must exist three horizontal lines and three vertical lines that intersect P', and each must contain exactly two points of P' of distinct colors. Moreover, no rainbow line can be horizontal or vertical. But this implies that out of the 9 intersections between horizontal and vertical lines, only 5 (the corners and the center) can be on a rainbow line. This contradicts $|P'| = 6$. ◀

We prove here a slightly stronger lower bound:

▶ **Theorem 10.** *Let $\mathcal{L} = \mathcal{L}_1 \cup \mathcal{L}_2 \cup \mathcal{L}_3 \cup \mathcal{L}_4$ be a 3-consistent colored set of lines in $\mathbb{R}^3$ with no colorful incidence and concurrent colors. If $|\mathcal{L}| < 24$, then $\mathcal{L}$ is contained in a 2-plane.*

Sketch of proof. We only outline our proof here and refer to [4] for the details. Assume, by contradiction, that $\mathcal{L}$ is a configuration with all required properties and $|\mathcal{L}| < 24$.

We first argue that $\mathcal{L}$ decomposes into a disjoint union of two colored sets of lines, each of which has 4 colors, is 3-consistent, has no colorful incidence, and has concurrent color classes. To do so, we consider the planes spanned by a line from the smallest color class, say $\mathcal{L}_1$, and the center of concurrence of another color class, say $\mathcal{L}_2$. The assumptions force every line of a color to intersect the lines of any other color in at least two points. This means that any such plane contains at least two lines of $\mathcal{L}_1$, so there are at most two planes. The same reason forces all lines from $\mathcal{L}_2$ to be contained in one plane or the other, and eventually the same goes for $\mathcal{L}_3$ and $\mathcal{L}_4$.

We then conclude by arguing that each of the subsets has at least 12 lines, forcing $\mathcal{L}$ to have at least 24 lines. This is straightforward if every color class has size at least 3. We argue that if a color class has size two, then all other color classes must have size at least 4. ◀

Classification. We also provide a characterization of 3-consistent, 4-colored sets of lines in $\mathbb{R}^3$ with no colorful incidence and 3 lines per color. Forgetting for a moment about colors, any such configuration must consist of 12 lines and 12 points, every point on 3 lines and every line through 3 points; in the classical tabulation of projective configurations, they are called (12_3) *configurations*. It turns out that there are 229 possible incidence structures meeting this description, and that every single one of them is realizable in $\mathbb{R}^3$ [9]. To analyze what happens when we add back the colors and the consistency assumption, we consider two special (12_3) configurations:

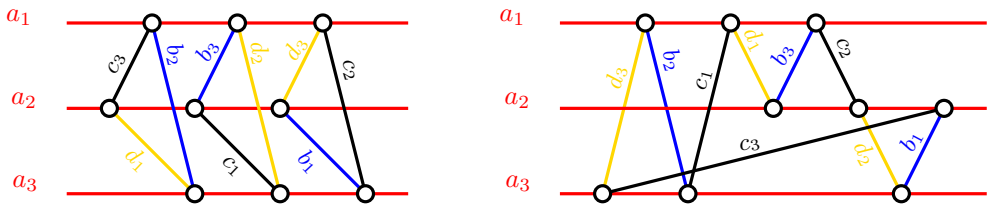
- A *Reye-type configuration*[7] is a configuration obtained by selecting 12 out of the 16 lines supporting the 12 edges and four long diagonals of a cube, in a way that produces a $(12)_3$ configuration.
- A *Desargues-type* configuration is defined from six planes $\Pi_1, \Pi_2, \ldots, \Pi_6$ in $\mathbb{R}^3$ where (i) each of $\{\Pi_1, \Pi_2, \ldots, \Pi_5\}$ and $\{\Pi_1, \Pi_2, \Pi_3, \Pi_6\}$ is in general position, and (ii) Π_4, Π_5 and Π_6 intersect in a line. The configuration consists of all lines that are contained in exactly two planes.

Here is our classification:

▶ **Theorem 11.** *Let $\mathcal{L} = \mathcal{L}_1 \cup \mathcal{L}_2 \cup \mathcal{L}_3 \cup \mathcal{L}_4$ be a 3-consistent colored set of lines in $\mathbb{R}^3$ with no colorful incidence. If every color class has size 3, and $\mathcal{L}$ is not contained in a 2-plane, then it is a Desargues-type or a Reye-type configuration colored as in Figure 2.*

Sketch of proof. We only outline our proof here, and refer to [4] for the details. The hypothesis imply that every line must intersect the lines of any other color in at least two points. This essentially allows us to establish that every color class consists of lines that are either pairwise skew, or concurrent and not coplanar. This geometric restriction implies, in turn, that for $i \neq j$, every line of $\mathcal{L}_i$ intersects exactly two lines of $\mathcal{L}_j$, and for any two lines of $\mathcal{L}_j$, there is exactly one line of $\mathcal{L}_i$ that intersects them both.

We use these two observations to reduce the sets of candidates for the incidence structure of the 12 lines. We fix a color class, say $A = \mathcal{L}_1$, and build a graph whose vertices are the 3-incidences involving a line of A, and where two vertices form an edge if the corresponding incidences have a line in common. This graph can be checked to be one of two candidates:



This already determines all 3-incidences that involve $\mathcal{L}_1$. The rest follows by observing that if the lines of $\mathcal{L}_1$ are pairwise skew (resp. concurrent and not coplanar) then two lines in $\mathcal{L}_2 \cup \mathcal{L}_3 \cup \mathcal{L}_4$ that intersect the same pair (resp. different pairs) of lines of $\mathcal{L}_1$ cannot intersect outside of $\mathcal{L}_1$. We end up with only two possible incidence structures (up to isomorphism, and possibly relabeling):

$$(I): \begin{array}{cccc} a_1b_2c_3 & a_1b_3d_2 & a_1c_2d_3 & b_1c_3d_2 \\ a_2b_3c_1 & a_2b_1d_3 & a_2c_3d_1 & b_2c_1d_3 \\ a_3b_1c_2 & a_3b_2d_1 & a_3c_1d_2 & b_3c_2d_1 \end{array} \quad \text{or} \quad (II): \begin{array}{cccc} a_1b_2c_3 & a_1b_3d_2 & a_1c_2d_3 & b_1c_1d_1 \\ a_2b_3c_1 & a_2b_1d_3 & a_2c_3d_1 & b_2c_2d_2 \\ a_3b_1c_2 & a_3b_2d_1 & a_3c_1d_2 & b_3c_3d_3. \end{array}$$

Figure 2 gives non-planar realizations of both set of incidences. We argue that these are essentially the only realizations by choosing a particular subset of points of incidence, and showing that their coordinates determines the whole geometric realization. This last step amounts, in each case, to an incidence theorem in projective geometry similar to the classic theorems of Reye or Desargues. ◀

[7] The $(12_4 16_3)$ configuration of Reye consists of 12 points and 16 lines in $\mathbb{R}^3$ such that every point is on 4 lines and every line contains 3 points; its realizations are projectively equivalent to the 16 lines supporting the 12 edges and four long diagonals of a cube, together with that cube's vertices and center and the 3 points at infinity in the directions of its edges

References

1 Kalle Åström, Roberto Cipolla, and Peter J Giblin. Generalised epipolar constraints. In *European Conference on Computer Vision*, pages 95–108. Springer, 1996.

2 Guillaume Batog, Xavier Goaoc, and Jean Ponce. Admissible linear map models of linear cameras. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1578–1585. IEEE, 2010.

3 Edmond Boyer. On using silhouettes for camera calibration. *Computer Vision–ACCV 2006*, pages 1–10, 2006.

4 Boris Bukh, Xavier Goaoc, Alfredo Hubard, and Matthew Trager. Consistent sets of lines with no colorful incidence. *Preprint arXiv:1803.06267*, 2018.

5 Paul Erdős and George Purdy. Some extremal problems in geometry. *Discrete Math*, pages 305–315, 1974.

6 Olivier Faugeras and Bernard Mourrain. On the geometry and algebra of the point and line correspondences between n images. In *Computer Vision, 1995. Proceedings., Fifth International Conference on*, pages 951–956. IEEE, 1995.

7 Jacob Fox, János Pach, Adam Sheffer, Andrew Suk, and Joshua Zahl. A semi-algebraic version of Zarankiewicz's problem. *Journal of the European Mathematical Society*, 19:1785–1810, 2017.

8 Ben Green and Terence Tao. On sets defining few ordinary lines. *Discrete & Computational Geometry*, 50(2):409–468, 2013.

9 Harald Gropp. Configurations and their realization. *Discrete Mathematics*, 174(1-3):137–151, 1997.

10 Larry Guth. Ruled surface theory and incidence geometry. In *A Journey Through Discrete Mathematics*, pages 449–466. Springer, 2017.

11 Larry Guth and Nets Hawk Katz. Algebraic methods in discrete analogs of the Kakeya problem. *Advances in Mathematics*, 225(5):2828–2839, 2010.

12 Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

13 Carlos Hernández, Francis Schmitt, and Roberto Cipolla. Silhouette coherence for camera calibration under circular motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(2), 2007.

14 Onur Özyeşil, Vladislav Voroninski, Ronen Basri, and Amit Singer. A survey of structure from motion. *Acta Numerica*, 26:305–364, 2017.

15 József Solymosi and Miloš Stojaković. Many collinear k-tuples with no $k+1$ collinear points. *Discrete & Computational Geometry*, 50(3):811–820, 2013.

16 Matthew Trager, Martial Hebert, and Jean Ponce. Consistency of silhouettes and their duals. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3346–3354, 2016.

17 Matthew Trager, Bernd Sturmfels, John Canny, Martial Hebert, and Jean Ponce. General models for rational cameras and the case of two-slit projections. In *CVPR 2017 - IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, United States, 2017. URL: `https://hal.archives-ouvertes.fr/hal-01506996`.

18 Oswald Veblen and John Wesley Young. *Projective geometry*, volume 1. Ginn, 1918.

The HOMFLY-PT Polynomial is Fixed-Parameter Tractable

Benjamin A. Burton
School of Mathematics and Physics, The University of Queensland
Brisbane QLD 4072, Australia
bab@maths.uq.edu.au

Abstract

Many polynomial invariants of knots and links, including the Jones and HOMFLY-PT polynomials, are widely used in practice but #P-hard to compute. It was shown by Makowsky in 2001 that computing the Jones polynomial is fixed-parameter tractable in the treewidth of the link diagram, but the parameterised complexity of the more powerful HOMFLY-PT polynomial remained an open problem. Here we show that computing HOMFLY-PT is fixed-parameter tractable in the treewidth, and we give the first sub-exponential time algorithm to compute it for arbitrary links.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability

Keywords and phrases Knot theory, knot invariants, parameterised complexity

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.18

Funding Supported by the Australian Research Council under the Discovery Projects scheme (DP150104108).

1 Introduction

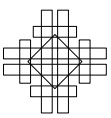
In knot theory, polynomial invariants are widely used to distinguish between different topological knots and links. Although they are powerful tools, these invariants are often difficult to compute: in particular, the one-variable Jones polynomial [10] and the stronger two-variable HOMFLY-PT polynomial [7, 20] are both #P-hard to compute in general [8, 22].

Despite this, we can use parameterised complexity to analyse classes of knots and links for which these polynomials become tractable to compute. In the early 2000s, as a part of a larger work on graph polynomials, Makowsky showed that the Jones polynomial can be computed in polynomial time for links whose underlying 4-valent graphs have bounded treewidth [15] – in other words, the Jones polynomial is *fixed-parameter tractable* with respect to treewidth.[1] A slew of other parameterised tractability results also appeared around this period for the Jones and HOMFLY-PT polynomials: parameters included the pathwidth of the underlying graph [16], the number of Seifert circles [16, 17], and the complexity of tangles in an algebraic presentation [16].

However, there was an important gap: it remained open as to whether the HOMFLY-PT polynomial is fixed-parameter tractable with respect to treewidth. This was dissatisfying because the HOMFLY-PT polynomial is both powerful and widely used, and the treewidth parameter lends itself extremely well to building fixed-parameter tractable algorithms, due to its strong connections to logic [4, 5] and its natural fit with dynamic programming.

The first major contribution of this paper is to resolve this open problem: we prove that computing the HOMFLY-PT polynomial of a link is fixed-parameter tractable with respect

[1] For an explicit algorithm, see Traldi [21].



to treewidth (Theorem 8). Our proof gives an explicit algorithm; this is feasible to implement, and will soon be released as part of the topological software package *Regina* [1, 2].

Regarding practicality: fixed-parameter tractable algorithms are only useful if the parameter is often small, and in this sense treewidth is a useful parameter: the underlying graph is planar, and so the treewidth of an n-crossing link diagram is at worst $O(\sqrt{n})$ [14]. This is borne out in practice – for instance, a simple greedy computation using *Regina* shows that, for Haken's famous 141-crossing "Gordian unknot", the treewidth is at most 12. Since HOMFLY-PT is a topological invariant, one can also attempt to use local moves on a link diagram to reduce the treewidth of the underlying graph, and *Regina* contains facilities for this also. More generally, Makowsky and Mariño [16] describe various classes of knots and links for which the treewidth is bounded.

There are few explicit algorithms in the literature for computing the HOMFLY-PT polynomial in general: the most notable is Kauffman's exponential-time *skein-template algorithm* [11], which forms the basis for our algorithm in this paper. Other notable algorithms are either designed for specialised inputs (e.g., Murakami et al.'s algorithm for 2-bridge links [18]), compute only portions of the HOMFLY-PT polynomial (e.g., Vertigan's polynomial-time algorithm for computing the first coefficients [19]), or are practical but do not prove unqualified guarantees on their complexity [3, 9],

The second major result of this paper is to improve the worst-case running time for computing the HOMFLY-PT polynomial in the general case, with no bound on the treewidth. In particular, we prove the first *sub-exponential* running time for arbitrary links. This is a simple corollary that follows immediately from analysing our fixed-parameter tractable algorithm using the $O(\sqrt{n})$ bound on the treewidth of a planar graph.

Throughout this paper we assume that the input link diagram contains no zero-crossing components (i.e., unknotted circles that are disjoint from the rest of the link diagram), since otherwise the number of crossings is not enough to adequately measure the input size. Such components are easy to handle – each zero-crossing component multiplies the HOMFLY-PT polynomial by $(\alpha - \alpha^{-1})z^{-1}$, and so we simply compute the HOMFLY-PT polynomial without them and then adjust the result accordingly.

2 Background

A *link* is a disjoint union of piecewise linear closed curves embedded in $\mathbb{R}^3$; the image of each curve is a *component* of the link. A *knot* is a link with precisely one component. In this paper we *orient* our links by assigning a direction to each component.

A *link diagram* is a piecewise linear projection of a link onto the plane, where the only multiple points are *crossings* at which one section of the link crosses another transversely. The sections of the link diagram between crossings are called *arcs*. The number of crossings is often used as a measure of input size; in particular, an n-crossing link diagram can be encoded in $O(n \log n)$ bits without losing any topological information.

Figure 1 shows two examples: the first is a knot with 4 crossings and 8 arcs, and the second is a 2-component link with 5 crossings and 10 arcs.

Figure 1 Examples of knots and links

Figure 2 Positive and negative crossings

Figure 3 Switching and splicing a crossing

Each crossing has a *sign* which is either *positive* or *negative*, according to the direction in which the upper strand passes over the lower; see Figure 2 for details. The *writhe* of a link diagram is the number of positive crossings minus the number of negative crossings (so the examples from Figure 1 have writhes 0 and -1 respectively).

In this paper we use two operations that change a link diagram at a single crossing. To *switch* a crossing is to move the upper strand beneath the lower, and to *splice* a crossing is to change the connections between the incoming and outgoing arcs; see Figure 3.

The *HOMFLY-PT polynomial* of a link $\mathcal{L}$ is a Laurent polynomial in the two variables α and z (a *Laurent polynomial* is a polynomial that allows both positive and negative exponents). There are two different but essentially equivalent definitions of the HOMFLY-PT polynomial in the literature (the other is typically given as a polynomial in ℓ and m [13]); we follow the same definition used by Kauffman [11].

A *parameterised problem* is a computational problem where the input includes some numerical parameter k. Such a problem is said to be *fixed-parameter tractable* if there is an algorithm with running time $O(f(k) \cdot \text{poly}(n))$, where f is an arbitrary function and n is the input size. A consequence of this is that, for any class of inputs whose parameter k is universally bounded, the algorithm runs in polynomial time.

Treewidth is a common parameter for fixed-parameter tractable algorithms on graphs, and we discuss it in detail now. Throughout this paper, all graphs are allowed to be multigraphs; that is, they may contain parallel edges and/or loops.

▶ **Definition 1** (Treewidth). Given a graph Γ with vertex set V, a *tree decomposition* of Γ consists of a tree τ and *bags* $\beta_i \subseteq V$ for each node i of T, subject to the following constraints: (i) each $v \in V$ belongs to some bag β_i; (ii) for each edge of Γ, its two endpoints $v, w \in V$ belong to some common bag β_i; and (iii) for each $v \in V$, the bags containing v correspond to a (connected) subtree of T.

The *width* of this tree decomposition is $\max |\beta_i| - 1$, and the *treewidth* of Γ is the smallest width of any tree decomposition of Γ.

A consequence of the Lipton-Tarjan planar separator theorem [14] is that every planar graph on n vertices has treewidth $O(\sqrt{n})$.

▶ **Definition 2** (Rooted tree decomposition). Let Γ be a graph. A *rooted tree decomposition* of Γ is a tree decomposition where one bag is singled out as the *root bag*. We define children and parents in the usual way: for any adjacent bags β, β' in the tree, if β is closer in the tree to the root than β' then we call β' a *child bag* of β, and we call β the (unique) *parent bag* of β'. A bag with no children is called a *leaf bag*.

More generally, we say that bag β' is a *descendant* of bag β if $\beta \neq \beta'$ and there is some sequence $\beta = \beta_0, \beta_1, \beta_2, \ldots, \beta_i = \beta'$ where each β_i is the parent bag of β_{i+1}.

▶ **Definition 3** (Nice tree decomposition)**.** Let Γ be a graph. A *nice tree decomposition* of Γ is a rooted tree decomposition with the following additional properties:

1. The root bag is empty.
2. Every leaf bag contains precisely one vertex.
3. Every non-leaf bag has either one or two child bags.
4. If a bag β_i has two child bags β_j and β_k, then $\beta_i = \beta_j = \beta_k$; we call β_i a *join bag*.
5. If a bag β_i has only one child bag β_j, then either:
   - $|\beta_i| = |\beta_j| + 1$ and $\beta_i \supset \beta_j$. Here we call β_i an *introduce bag*, and the single vertex in $\beta_i \backslash \beta_j$ is called the *introduced vertex*.
   - $|\beta_i| = |\beta_j| - 1$ and $\beta_i \subset \beta_j$. Here we call β_i a *forget bag*, and the single vertex in $\beta_j \backslash \beta_i$ is called the *forgotten vertex*.

3 Kauffman's skein-template algorithm

Kauffman's skein-template algorithm works by building a decision tree. The leaves of this decision tree are obtained from the original link by switching and/or splicing some crossings. Each leaf is then evaluated as a Laurent polynomial, according to the number of components and the specific switches and/or splices that were performed, and these are summed to obtain the final HOMFLY-PT polynomial.

Our fixed-parameter tractable algorithm (described in Section 4) works by inductively constructing, aggregating and analysing small pieces of Kauffman's decision tree. We therefore devote this section to describing Kauffman's algorithm in detail, beginning with a description of the algorithm itself followed by a detailed example.

▶ **Algorithm 4** (Kauffman [11])**.** Let $\mathcal{L}$ be a link diagram with n crossings (and therefore $2n$ arcs). Then the following procedure computes the HOMFLY-PT polynomial of $\mathcal{L}$.

Arbitrarily label the arcs $1, 2, \ldots, 2n$. We build a decision tree by walking through the link as follows:

- Locate the lowest-numbered arc that has not yet been traversed, and follow the link along this arc in the direction of its orientation.
- Each time we encounter a new crossing that has not yet been traversed:
  - If we are passing *over* the crossing, then we simply pass through it and continue traversing the link.
  - If we are passing *under* the crossing, then we make a fork in the decision tree. On one branch we *splice* the crossing, and on the other branch we *switch* the crossing. Either way, we then pass through the crossing (following the splice if we made one) and continue traversing the link.
- Whenever we encounter a crossing for the second time – regardless of whether it was first seen on the over or under strand – we simply pass through it (again following the splice if we made one) and continue traversing the link.
- Whenever we return to an arc that has already been traversed (thus closing off a component of our modified link):
  - If there are still arcs remaining that have not yet been traversed, then we locate the lowest-numbered such arc and continue our traversal from there.
  - If every arc has now been traversed, then the resulting modified link becomes a leaf of our decision tree.

To each leaf of the decision tree, we assign the polynomial term $(-1)^{t_-} z^t \alpha^{w-w_0} \delta^{c-1}$, where:

- t is the number of splices that we performed on this branch of the decision tree;
- t_- is the number of splices that we performed on negative crossings;

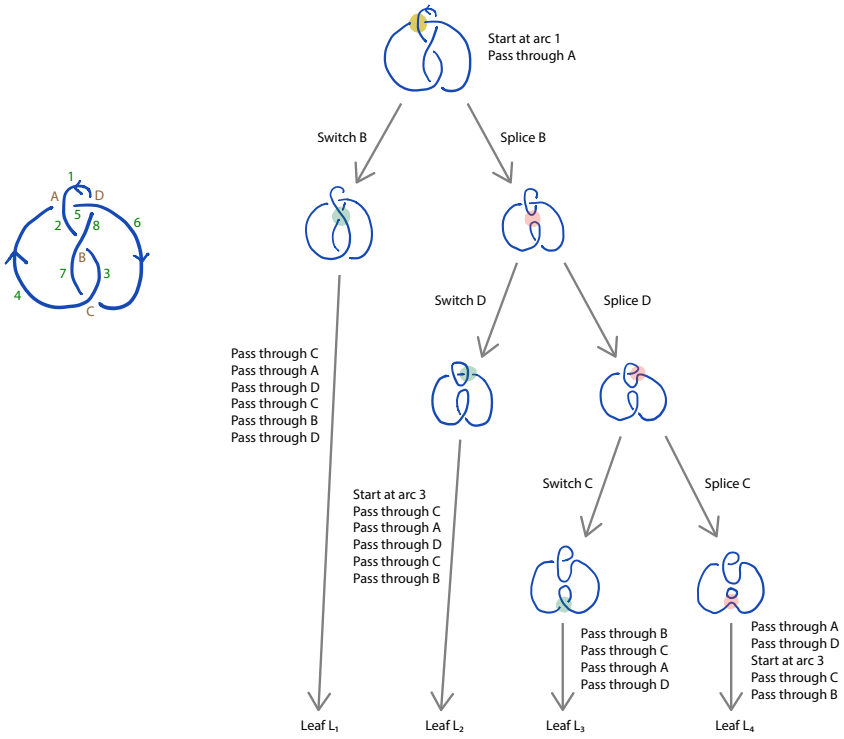


Figure 4 Running Kauffman's skein-template algorithm

- w is the writhe of the modified link, after any switching and/or splicing;
- w_0 is the writhe of the original link $\mathcal{L}$, before any switching or splicing;
- c is the number of components of the modified link;
- δ expands to the polynomial $(\alpha - \alpha^{-1})z^{-1}$.

The HOMFLY-PT polynomial of $\mathcal{L}$ is then the sum of these polynomial terms over all leaves.

Note that different branches of the decision tree may traverse the arcs of the link in a different order, since each splice changes the connections between arcs; likewise, the modified links at the leaves of the decision tree may have different numbers of link components.

▶ **Example 5.** Figure 4 shows the algorithm applied to the figure eight knot, as depicted at the root of the tree. The eight arcs are numbered 1–8; to help with the discussion we also label the four crossings A, B, C and D, which have signs $+$, $-$, $-$ and $+$ respectively.

We begin at arc 1, and because we first encounter crossing A on the upper strand, we leave it unchanged and move on to arc 2. For crossing B we can either switch or splice, and in these cases the traversal continues to arc 3 or 8 respectively. The decision process continues as shown in the diagram, resulting in the four leaves L_1, L_2, L_3 and L_4.

Of particular note is the branch where we splice B and then switch D. Here the traversal runs through arcs 1, 2 and 8, at which point it returns to arc 1, closing off a small loop. We now begin again at arc 3: this takes us through crossing C (which we pass through because we see it first on the upper strand), then crossing A (which we pass through because we are seeing it for the second time), then crossing D (which we likewise pass through), and so on.

The polynomials assigned to the four leaves are shown below:

	t	t_-	w	$w-w_0$	c	Poly.
L_1	0	0	2	2	1	α^2
L_2	1	1	-1	-1	2	$-z\alpha^{-1}\delta$

	t	t_-	w	$w-w_0$	c	Poly.
L_3	2	1	2	2	1	$-z^2\alpha^2$
L_4	3	2	1	1	2	$z^3\alpha\delta$

This yields the final HOMFLY-PT polynomial

$$\alpha^2 - z^2\alpha^2 + (z^3\alpha - z\alpha^{-1})\delta = \alpha^2 - z^2\alpha^2 + (z^3\alpha - z\alpha^{-1})(\alpha - \alpha^{-1})z^{-1} = \alpha^2 + \alpha^{-2} - z^2 - 1.$$

▶ **Theorem 6.** *Kauffman's skein-template algorithm computes the HOMFLY-PT polynomial of an n-crossing link in time $2^n \cdot \mathrm{poly}(n)$.*

Proof. The decision tree has $\leq 2^n$ leaves, since we only branch the first time we traverse each crossing (and even then, only if we first traverse the crossing from beneath, not above). All other operations are polynomial time, giving an overall running time of $2^n \cdot \mathrm{poly}(n)$ ◀

Although it requires exponential time, Kauffman's algorithm can compute the HOMFLY-PT polynomial in polynomial *space*. This is because we do not need to store the entire decision tree – we can simply perform a depth-first traversal through the tree, making and undoing switches and splices as we go, and keep a running total of the polynomial terms for those leaves that we have encountered so far.

4 A fixed-parameter tractable algorithm

In this section we present an explicit algorithm to show that computing the HOMFLY-PT polynomial is fixed-parameter tractable in the treewidth of the input link diagram.

▶ **Definition 7.** Let $\mathcal{L}$ be a link diagram. The *graph of* $\mathcal{L}$, denoted $\Gamma(\mathcal{L})$, is the directed planar 4-valent multigraph whose vertices are the crossings of $\mathcal{L}$, and whose directed edges are the oriented arcs of $\mathcal{L}$.

The first main result of this section, which resolves the open problem of the parameterised complexity of computing the HOMFLY-PT polynomial, is:

▶ **Theorem 8.** *Consider the parameterised problem whose input is a link diagram $\mathcal{L}$, whose parameter is the treewidth of the graph $\Gamma(\mathcal{L})$, and whose output is the HOMFLY-PT polynomial of $\mathcal{L}$. Then this problem is fixed-parameter tractable.*

4.1 Algorithm overview

The remainder of Section 4 is devoted to proving Theorem 8. First, however, we give a brief overview of the algorithm and the difficulties that it must overcome.

Roughly speaking, our algorithm takes a nice tree decomposition of $\Gamma(\mathcal{L})$ and works from the leaf bags to the root bag. For each bag of the tree, we consider a range of possible "boundary conditions" for how a link traversal interacts with the bag, and for each set of boundary conditions we aggregate all "partial leaves" of Kauffman's decision tree that satisfy them. We formalise these boundary conditions and their resulting aggregations using the notions of a *configuration* and *evaluation* respectively (Definitions 12 and 16).

This general pattern of dynamic programming over a tree decomposition is common for fixed-parameter tractable algorithms. The main difficulty that we must overcome is to keep the number of configurations polynomial in the number of crossings n. This difficulty arises

because the choices in Kauffman's decision tree depend upon the *order* in which you traverse the crossings, and so each configuration must encode a starting arc for every connected portion of a link traversal in every "partial leaf" of the decision tree. Because a "partial leaf" could contain $O(n)$ disjoint portions of a traversal, each with $O(n)$ potential starting arcs, the resulting number of configurations would grow at a rate of $O(n^n)$, which is too large.

Our solution is the following. Recall that Kauffman's algorithm uses an arbitrary ordering of the arcs of the link to determine the order in which we traverse arcs and make decisions (to pass through, switch and/or splice crossings). In our algorithm, we order the arcs using the tree decomposition – for each directed arc, we identify the forget bag in which its end crossing is forgotten, and we then order the arcs according to how close this forget bag is to the root of the tree. This makes the ordering of arcs *inherent* to the tree decomposition, and so we do not need to explicitly encode starting arcs in our configurations. This is enough to reduce the number of configurations at each bag to a function of the treewidth alone, with no dependency on n.

4.2 Properties of tree decompositions

We now make some small observations about tree decompositions of the graphs of links.

▶ **Definition 9.** Let $\mathcal{L}$ be a link diagram, let T be a rooted tree decomposition of $\Gamma(\mathcal{L})$, and let β be any bag of T. For each crossing c of $\mathcal{L}$, we say that:

- c is *unvisited* at β if c does not appear in either β or any bags in the subtree rooted at β;
- c is *current* at β if c appears in the bag β itself;
- c is *forgotten* at β if c does not appear in the bag β, but does appear in some bag in the subtree rooted at β.

Observe that the unvisited, current and forgotten crossings at β together form a *partition* of all crossings of $\mathcal{L}$.

▶ **Lemma 10.** *Let $\mathcal{L}$ be a link diagram, let T be a rooted tree decomposition of $\Gamma(\mathcal{L})$, and let β be any bag of T. Then no arc of $\mathcal{L}$ can connect a crossing that is forgotten at β with a crossing that is unvisited at β, or vice versa.*

Proof. Let crossing c be forgotten at β, and let crossing d be unvisited at β. If there were an arc from c to d (or vice versa) then some bag of T would need to contain both c and d, by condition (ii) of Definition 1.

Since c appears in a descendant bag of β but not β itself, condition (iii) of Definition 1 means that *all* bags containing c must be descendant bags of β. However, since d is unvisited, no bag containing d can be a descendant bag of β, yielding a contradiction. ◀

▶ **Lemma 11.** *Let $\mathcal{L}$ be a link diagram, let T be a nice tree decomposition of $\Gamma(\mathcal{L})$, and let c be any crossing of $\mathcal{L}$. Then T has a unique forget bag for which c is the forgotten vertex.*

Proof. Since the root bag of T is empty, there must be some forget bag for which c is the forgotten vertex. Moreover, since the bags containing c form a subtree of T, there is only one bag that contains c but whose parent does not – the root of this subtree. ◀

4.3 Framework for the algorithm

We now define precisely the problems that we solve at each stage of the algorithm. Our first task is to define a *configuration* – that is, the "boundary conditions" that describe how a link traversal interacts with an individual bag of the tree decomposition.

▶ **Definition 12.** Let $\mathcal{L}$ be a link diagram, let T be a rooted tree decomposition of $\Gamma(\mathcal{L})$, and let β be any bag of T. Then a *configuration* at β is a sequence of the form $(a_1, b_1, a_2, b_2, \ldots, a_u, b_u)$, where:

1. Each a_i is an outgoing arc from some crossing that is current at β, where the destination of this arc is a crossing that is either current or forgotten at β. Moreover, every such arc must appear as exactly one of the a_i.
2. Each b_i is an incoming arc to some crossing that is current at β, where the source of this arc is a crossing that is either current or forgotten at β. Moreover, every such arc must appear as exactly one of the b_i.
3. If an arc of $\mathcal{L}$ connects two crossings that are *both* current at β, then by conditions 1 and 2, such an arc must appear as some a_i and also as some b_j. In this case we also require that $i = j$.

We call each pair a_i, b_i a *matching pair* of arcs in the configuration, and if $a_i = b_i$ (as in condition 3 above) then we call this a *trivial pair*.

Intuitively, each matching pair a_i, b_i describes the start and end points of a "partial traversal" of the link (possibly after some switches and/or splices) that starts and ends in the bag β, and that *only passes through forgotten crossings*. By placing these endpoints in a sequence $a_1, b_1, \ldots, a_u, b_u$, we impose an ordering upon these "partial traversals" (which we will eventually use to order the traversal of arcs in Kauffman's decision tree).

▶ **Lemma 13.** *Let $\mathcal{L}$ be a link diagram, let T be a rooted tree decomposition of $\Gamma(\mathcal{L})$, and let β be any bag of T. Then configurations at β exist (i.e., the definition above can be satisfied). Moreover, then there are at most $(2|\beta|)!^2$ possible configurations at β.*

Proof. To show that the definition can be satisfied, all we need to show is that the number of arcs from a current crossing to a current-or-forgotten crossing (i.e., the number of arcs a_i) equals the number of arcs from a current-or-forgotten crossing to a current crossing (i.e., the number of arcs b_i). This follows immediately from the fact that there are no arcs joining a forgotten crossing with an *unvisited* crossing (Lemma 10).

The number of configuration is a simple exercise in counting: there are exactly $|\beta|$ crossings current at β, each with exactly two outgoing and two incoming arcs. This yields at most $2|\beta|$ arcs a_i and $2|\beta|$ arcs b_j, giving at most $(2|\beta|)!$ possible orderings of the a_i and $(2|\beta|)!$ possible orderings of the b_i. ◀

Our next task is to describe how we order the arcs in Kauffman's decision tree in order to avoid having to explicitly track the start points of link traversals in our algorithm.

▶ **Definition 14.** Let $\mathcal{L}$ be a link diagram, and let T be a nice tree decomposition of $\Gamma(\mathcal{L})$.

Let $a_1, \ldots, a_{2n}$ be the directed arcs of $\mathcal{L}$. Let c_i denote the crossing at the end of arc a_i, and let β_i be the (unique) forget bag that forgets the crossing c_i.

A *tree-based ordering* of the arcs of $\mathcal{L}$ is a total order on the arcs $\{a_i\}$ that follows a depth-first ordering of the corresponding bags $\{\beta_i\}$. That is:

1. whenever β_i is a descendant bag of β_j, we must have $a_j < a_i$;
2. for any two disjoint subtrees of T, *all* of the arcs whose corresponding bags appear in one subtree must be ordered before *all* of the arcs whose corresponding bags appear in the other subtree.

Essentially, this orders the arcs of $\mathcal{L}$ according to how close to the root of T their ends are forgotten – arcs are ordered *earlier* when the crossings they point to are forgotten *closer*

to the root. There are many such possible orderings; for our algorithm, any one will do.[2]

We now proceed to define an *evaluation* – that is, the aggregation that we perform for each configuration at each bag.

▶ **Definition 15.** Let $\mathcal{L}$ be a link diagram and let T be a nice tree decomposition of $\Gamma(\mathcal{L})$. Fix a tree-based ordering $<$ of the arcs of $\mathcal{L}$, and let $\kappa = (a_1, b_1, a_2, b_2, \ldots, a_u, b_u)$ be a configuration at some bag β.

A *partial leaf* for κ assigns one of the three tags `pass`, `switch` or `splice` to each forgotten crossing at β, under the following conditions.

Consider (i) all the forgotten *crossings* of $\mathcal{L}$, after applying any switches and/or splices as described by the chosen tags; and (ii) all the *arcs* of $\mathcal{L}$ whose two endpoints are forgotten and/or current. These join together to form a collection of (i) connected segments of a link that start and end at current crossings and whose intermediate crossings are all forgotten; and (ii) closed components of a link that contain only forgotten crossings. We require that:

1. Each segment (as opposed to a closed component) must begin at some arc a_i and end at the matching arc b_i.
2. Suppose we traverse the segments and closed components in the following order. First we traverse the segments in the order described by κ (i.e., the segment from a_1 to b_1, then from a_2 to b_2, and so on). Then we traverse the closed components according to the ordering $<$: we find the closed component with the smallest arc according to $<$ and traverse it starting at that arc; then we find the closed component with the smallest arc not yet traversed and traverse it from that arc; and so on.

   Then the `pass`, `switch` and `splice` tags that we assign to each forgotten crossing must be consistent with Kauffman's algorithm under this traversal. Specifically:
   - If we encounter a forgotten crossing for the first time on the upper strand, then it must be marked `pass`.
   - If we encounter a forgotten crossing for the first time on the lower strand, then it must be marked either `switch` or `splice`.

This definition appears complex, but in essence, a partial leaf for κ is simply a choice of operations on each forgotten crossing that *could* eventually be extended to a leaf of the decision tree in Kauffman's original algorithm.

Note that the order of traversal in condition 2 is indeed consistent with Kauffman's algorithm. The segments must be traversed before the closed components; this is because the segments will be extended and eventually closed off as the algorithm moves towards the root of the tree, and so the segments will eventually contain arcs that are smaller (according to $<$) than any of the arcs in the closed components in condition 2 above.

▶ **Definition 16.** Let $\mathcal{L}$ be a link diagram and let T be a nice tree decomposition of $\Gamma(\mathcal{L})$. Fix a tree-based ordering $<$ of the arcs of $\mathcal{L}$, and let κ be a configuration at some bag β.

The *evaluation* of κ is a Laurent polynomial in the variables α, z and δ, obtained by summing the terms $(-1)^{t_-} z^t \alpha^{w-w_0} \delta^{c-1}$ over all partial leaves for κ, where:

- t is the number of forgotten crossings marked `splice`;
- t_- is the number of forgotten negative crossings marked `splice`;

[2] Different tree-based orderings share many common properties. For example, given any collection of arcs that are connected in $\Gamma(\mathcal{L})$, all tree-based orderings share the same *minimum* arc in this collection. This is enough to ensure that different tree-based orderings will traverse the arcs and crossings of $\mathcal{L}$ in *exactly* the same order when running Kauffman's algorithm.

- w is the number of forgotten positive crossings minus the number of forgotten negative crossings, where we ignore any crossings marked `splice` and we reverse the sign of any crossings marked `switch`;
- w_0 is the writhe of the entire original link diagram $\mathcal{L}$ (including all crossings);
- c is the number of closed components that contain only forgotten crossings, as described in condition 2 of Definition 15.

The structure of the algorithm itself is now simple: we move through the tree decomposition from the leaf bags to the root bag, and at each bag β we compute the evaluation of all configurations at β.

This process eventually ends at the root bag, where we can show that the evaluation of the (unique) empty configuration encodes the HOMFLY-PT polynomial of the link $\mathcal{L}$:

▶ **Lemma 17.** *Let $\mathcal{L}$ be a link diagram and let T be a nice tree decomposition of $\Gamma(\mathcal{L})$. Fix a tree-based ordering $<$ of the arcs of $\mathcal{L}$.*

Then there is only one configuration at the root bag of T (which is the empty sequence). Moreover, if the evaluation of this configuration is the Laurent polynomial $Q(\alpha, z, \delta)$, then the HOMFLY-PT polynomial of $\mathcal{L}$ is obtained by replacing δ with $(\alpha - \alpha^{-1})z^{-1}$.

Proof. At the root bag, *every* crossing is forgotten; therefore no crossings are current and so the only configuration is the empty sequence. Call this κ_0.

Following Definition 15, we then see that the partial leaves for κ_0 are precisely the leaves of the decision tree in Kauffman's skein-template algorithm, assuming that we order the arcs in Kauffman's algorithm using our tree-based ordering $<$.

Moreover, when evaluating κ_0, the terms $(-1)^{t-} z^t \alpha^{w-w_0} \delta^{c-1}$ that we sum are precisely the polynomials that we sum in Kauffman's algorithm, with the exception that we keep δ as a separate variable instead of expanding it to $(\alpha - \alpha^{-1})z^{-1}$.

It follows that, if we take this evaluation and expand δ to $(\alpha - \alpha^{-1})z^{-1}$, then we obtain the same result as Kauffman's algorithm, which is the HOMFLY-PT polynomial of $\mathcal{L}$. ◀

4.4 Running the algorithm

Having defined the problems to solve at each bag, we can now describe the algorithm in full.

▶ **Algorithm 18.** Suppose we are given a link diagram $\mathcal{L}$ and a nice tree decomposition T of $\Gamma(\mathcal{L})$. Then the following algorithm computes the HOMFLY-PT polynomial of $\mathcal{L}$.

If $\mathcal{L}$ contains any trivial twists – that is, arcs that run from a crossing back to itself – then we untwist them now. This preserves the topology of the link, and so does not change the HOMFLY-PT polynomial. If this produces any zero-crossing components then we also remove them – this *does* change the HOMFLY-PT polynomial (as explained in the introduction, we lose a factor of $(\alpha - \alpha^{-1})z^{-1}$), but we can simply adjust the result once the algorithm has finished by multiplying through by $(\alpha - \alpha^{-1})z^{-1}$ again.

Next, fix a tree-based ordering $<$ of the arcs of $\mathcal{L}$.

Now, as described at the end of Section 4.3, we work through the bags of T in order from leaves to root. At each bag β we compute and store the evaluation of all configurations at β. How we do this depends upon the type of the bag β.

If β is a leaf bag:

In this case we have exactly one current crossing c, and every incoming and outgoing arc from c connects it to an unvisited crossing. Therefore there is only one configuration (the empty sequence). Moreover, since there are no forgotten crossings at all, this configuration has an evaluation of $\alpha^{-w_0}\delta^{-1}$, where w_0 is the writhe of the entire input diagram $\mathcal{L}$.

If β is an introduce bag:

Let c be the new crossing that is introduced in β, and let β' be the child bag of β. Note that, by applying Lemma 10 to the bag β', we see that each of the four arcs that meets c must connect c to either a current or unvisited crossing at β.

If all four of these arcs connect c to an unvisited crossing at β, then the configurations at β are precisely the configurations at β'. Moreover, since the forgotten crossings at β' and β are the same, it follows that the partial leaves and evaluation of each configuration will be identical at bags β' and β, and so we can copy all of our computations from the child bag β' directly to β with no changes.

If one or more arcs connects c to a current crossing at β, then each configuration κ' at β' gives rise to many configurations at β. Specifically, each such arc a will appear as a new trivial pair $a_i = b_i = a$ in the sequence (see condition 3 of Definition 12). This pair may be inserted anywhere amongst the matching pairs from κ'; that is, we can extend the sequence $(a_1, b_1, \ldots, a_u, b_u)$ to $(a_1, b_1, \ldots, a_j, b_j, a, a, a_{j+1}, b_{j+1}, \ldots, a_u, b_u)$ for any insertion point j. As before, the partial leaves after this insertion are identical to the partial leaves for κ', and so the evaluation of each such new configuration is identical to the evaluation of κ'.

If β is a join bag:

Let β_1 and β_2 be the two child bags of β. We iterate through all pairs (κ_1, κ_2) where each κ_i is a configuration at β_i, and attempt to find "compatible" pairs that can be merged to form a configuration at β. Note that all forgotten crossings at β_1 will be unvisited at β_2, and all forgotten crossings at β_2 will be unvisited at β_1.

The only arcs that appear in the sequences for *both* κ_1 and κ_2 are those arcs whose endpoints are both current at β. By Definition 12, such arcs must appear as trivial pairs in both κ_1 and κ_2. Therefore, if these trivial pairs all appear in the same relative order in both κ_1 and κ_2, we can merge κ_1 and κ_2 to form a configuration at β – in fact there are many ways to do this, since we can interleave the two sequences for κ_1 and κ_2 however we like as long as the matching pairs from each individual κ_i are all kept in the same relative order.

Since the forgotten crossings for β_1 and β_2 are disjoint, we can combine any partial leaf for κ_1 with any partial leaf for κ_2 to form a partial leaf for the new configuration κ at β. Therefore the evaluation of κ is $e_1 \cdot e_2 \cdot \alpha^{w_0}\delta$, where each e_i is the evaluation of κ_i. Here the extra factor of $\alpha^{w_0}\delta$ compensates for the fact that each polynomial term from Definition 16 includes a "constant factor" of $\alpha^{-w_0}\delta^{-1}$ which we inherit twice from e_1 and e_2.

If the trivial pairs for κ_1 and κ_2 do *not* appear in the same relative order in both sequences, then we cannot merge the two configurations to form a new configuration at β, and so we ignore this pair of configurations (κ_1, κ_2) and move on to the next pair.

If β is a forget bag:

Let c be the crossing that is forgotten in β, and let β' be the child bag of β. Once more we iterate through all configurations at β'; let κ' be such a configuration.

We consider applying each of the tags `pass`, `switch` and `splice` to the forgotten crossing c. For consistency with Kauffman's decision tree, we only allow the `pass` tag if the upper incoming arc into c appears earlier in κ' than the lower incoming arc into c (which means we first encounter c on the upper strand); likewise, we only allow the `switch` and `splice` tags if the lower incoming arc into c appears earlier in κ' than the upper incoming arc into c.

Having chosen a tag, we then attempt to convert κ' into a new configuration κ at β. This involves combining matching pairs of κ' that connect with c to reflect how the link traversal passes through the now-forgotten crossing c. There are two ways that this can be done:

- Matching pairs on either side of c could be adjacent in κ'. For instance, suppose we apply the `switch` tag. Then κ' could be of the form $\ldots, a_i, b_i, a_{i+1}, b_{i+1}, \ldots$, where b_i is an incoming arc for c and a_{i+1} is the opposite outgoing arc for c (in the case of `splice`, a_{i+1} would need to be the *adjacent* outgoing arc instead). The new configuration κ would then be $\ldots, a_i, b_{i+1}, \ldots$; here (a_i, b_{i+1}) becomes a new matching pair.
- There could be a single matching pair in κ' that runs from c back around to itself. For instance, if we apply the `switch` tag then κ' could be of the form $\ldots, a_i, b_i, \ldots$, where a_i is an outgoing arc for c and b_i is the opposite incoming arc (again, for `splice` we would need b_i to be the *adjacent* incoming arc instead). In this case, forgetting c will connect the two ends of the traversal segment from a_i to b_i to form a new closed link component, and the new configuration κ is obtained by deleting the pair (a_i, b_i) from κ'.

Note that we must combine *two* pairs of matching pairs – one for each strand that passes through c. If this cannot be done as described above (i.e., the relevant matching pairs are neither adjacent in κ' nor do they run from c back to itself), then we cannot apply our chosen tag to κ'. In this case we just move to the next choice of tag for c and/or the next available configuration at β'.

If we *are* able to use our chosen tag with κ', then we can use the evaluation of κ' to compute the evaluation of the new configuration κ. We must, however, multiply by an appropriate factor to reflect how the partial leaves have changed, following Definition 16:

- if we chose `splice` then we must multiply by z, and also by -1 if c is a negative crossing;
- if we `pass` a positive crossing or `switch` a negative crossing, we must multiply by α;
- if we `pass` a negative crossing or `switch` a positive crossing, we must multiply by α^{-1};
- if we formed a new closed link component then we must multiply by δ.

Since κ is obtained by deleting and/or merging matching pairs from κ', it is possible that several different child configurations κ' could yield the *same* new configuration κ. If this happens, we simply sum all of the resulting evaluations at κ.

Once we have finished working through all the bags, we take the evaluation of the unique configuration at the root bag and expand δ to $(\alpha - \alpha^{-1})z^{-1}$ as described in Lemma 17. This yields the final HOMFLY-PT polynomial of $\mathcal{L}$.

▶ **Theorem 19.** *If the nice tree decomposition in Algorithm 18 has $O(n)$ bags and width k, then the algorithm has running time $O\left((2k)!^4 \cdot \text{poly}(n)\right)$, where n is the number of crossings in the link diagram $\mathcal{L}$.*

Proof. Most of the operations in the algorithm are clearly polynomial time, and we do not discuss their precise complexities here. The only source of super-polynomial running time comes from the large number of configurations to process at each bag.

When processing a forget or introduce bag, Lemma 13 shows that there are $\leq (2k)!^2$ child configurations to process, requiring $O\left((2k)!^2 \cdot \text{poly}(n)\right)$ time in total. When processing a join bag, we iterate through *pairs* of configurations (κ_1, κ_2), and so the total processing time becomes $O\left((2k)!^4 \cdot \text{poly}(n)\right)$. Note that, although any individual pair (κ_1, κ_2) could yield a super-polynomial number of new configurations κ (due to the many possible ways to merge configurations), these nevertheless contribute to a *total* number of configurations at the join bag which is still bounded by Lemma 13, and so the total processing time at a join bag remains no worse than $O\left((2k)!^4 \cdot \text{poly}(n)\right)$. ◀

Algorithm 18 assumes that you already have a tree decomposition; however, finding one with the smallest possible width is an NP-hard problem [6]. We therefore extend our running time analysis to the more common case where only the link is given, and a tree decomposition is *not* known in advance.

▶ **Corollary 20.** *Given a link diagram $\mathcal{L}$ with n crossings whose graph $\Gamma(\mathcal{L})$ has treewidth k, it is possible to compute the HOMFLY-PT polynomial of $\mathcal{L}$ in time $O\left((8k)!^4 \cdot \mathrm{poly}(n)\right)$.*

Proof. Cygan et al. [6] give an algorithm that can construct a tree decomposition with width $\leq 4k+4$ and n bags in time $O(8^k k^2 \cdot n^2)$. Kloks [12] then shows how to convert this into a *nice* tree decomposition in $O(n)$ time with the same width, and with $O(n)$ bags. Our corollary now follows by applying Theorem 19 with width $4k+4$. Note that the running time from Theorem 19 dominates the preprocessing time required to build the tree decompositions. ◀

Corollary 20 shows that computing the HOMFLY-PT polynomial is fixed-parameter tractable, thereby finally concluding the proof of Theorem 8, our first main result.

However, unlike Kauffman's algorithm, our algorithm is *not* polynomial space, since it must store up to $(2k)!^2$ configurations and their evaluations at each bag.

We can now finish this paper with our second main result. Since the treewidth of a planar graph is $O(\sqrt{n})$, we can substitute $k = O(\sqrt{n})$ into Corollary 20 to yield the following:

▶ **Corollary 21.** *Given a link diagram $\mathcal{L}$ with n crossings, it is possible to compute the HOMFLY-PT polynomial of $\mathcal{L}$ in time $e^{O(\sqrt{n} \cdot \log n)}$.*

That is, it is possible to compute the HOMFLY-PT polynomial in sub-exponential time.

References

1. Benjamin A. Burton. Introducing Regina, the 3-manifold topology software. *Experiment. Math.*, 13(3):267–272, 2004.
2. Benjamin A. Burton, Ryan Budney, William Pettersson, et al. Regina: Software for low-dimensional topology. `http://regina-normal.github.io/`, 1999–2018.
3. Federico Comoglio and Maurizio Rinaldi. A topological framework for the computation of the HOMFLY polynomial and its application to proteins. *PLoS ONE*, 6(4):e18693, 2011.
4. Bruno Courcelle. On context-free sets of graphs and their monadic second-order theory. In *Graph-Grammars and their Application to Computer Science (Warrenton, VA, 1986)*, volume 291 of *Lecture Notes in Comput. Sci.*, pages 133–146. Springer, Berlin, 1987.
5. Bruno Courcelle. Graph rewriting: An algebraic and logic approach. In *Handbook of Theoretical Computer Science, Vol. B*, pages 193–242. Elsevier, Amsterdam, 1990.
6. Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, Cham, 2015.
7. P. Freyd, D. Yetter, J. Hoste, W. B. R. Lickorish, K. Millett, and A. Ocneanu. A new polynomial invariant of knots and links. *Bull. Amer. Math. Soc. (N.S.)*, 12(2):239–246, 1985.
8. F. Jaeger, D. L. Vertigan, and D. J. A. Welsh. On the computational complexity of the Jones and Tutte polynomials. *Math. Proc. Cambridge Philos. Soc.*, 108(1):35–53, 1990.
9. Robert J. Jenkins, Jr. *A Dynamic Programming Approach to Calculating the HOMFLY Polynomial for Directed Knots and Links*. Master's thesis, 1989.
10. Vaughan F. R. Jones. A polynomial invariant for knots via von Neumann algebras. *Bull. Amer. Math. Soc. (N.S.)*, 12(1):103–111, 1985.
11. Louis H. Kauffman. State models for link polynomials. *Enseign. Math. (2)*, 36(1-2):1–37, 1990.
12. Ton Kloks. *Treewidth: Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1994.
13. W. B. Raymond Lickorish. *An Introduction to Knot Theory*. Number 175 in Graduate Texts in Mathematics. Springer, New York, 1997.

14 Richard J. Lipton and Robert Endre Tarjan. A separator theorem for planar graphs. *SIAM J. Appl. Math.*, 36(2):177–189, 1979.

15 J. A. Makowsky. Coloured Tutte polynomials and Kauffman brackets for graphs of bounded tree width. *Discrete Appl. Math.*, 145(2):276–290, 2005.

16 J. A. Makowsky and J. P. Mariño. The parameterized complexity of knot polynomials. *J. Comput. Syst. Sci.*, 67(4):742–756, 2003.

17 H. R. Morton and H. B. Short. Calculating the 2-variable polynomial for knots presented as closed braids. *J. Algorithms*, 11(1):117–131, 1990.

18 Masahiko Murakami, Fumio Takeshita, and Seiichi Tani. Computing HOMFLY polynomials of 2-bridge links from 4-plat representation. *Discrete Appl. Math.*, 162:271–284, 2014.

19 Józef H. Przytycki. The first coefficient of Homflypt and Kauffman polynomials: Vertigan proof of polynomial complexity using dynamic programming. Preprint, `arXiv:1707.07733`, 2017.

20 Józef H. Przytycki and Paweł Traczyk. Invariants of links of Conway type. *Kobe J. Math.*, 4(2):115–139, 1988.

21 L. Traldi. On the colored Tutte polynomial of a graph of bounded tree-width. *Discrete Applied Mathematics*, 154.6:1032–1036, 2006.

22 D. J. A. Welsh. *Complexity: Knots, Colourings and Counting*, volume 186 of *London Math. Soc. Lecture Note Ser.* Cambridge Univ. Press, Cambridge, 1993.

Practical Volume Computation of Structured Convex Bodies, and an Application to Modeling Portfolio Dependencies and Financial Crises*

Ludovic Calès[1]
European Commission, Joint Research Centre, Ispra, Italy
ludovic.cales@ec.europa.eu

Apostolos Chalkis[2]
Department of Informatics & Telecommunications
National & Kapodistrian University of Athens, Greece
achalkis@di.uoa.gr

Ioannis Z. Emiris[2]
Department of Informatics & Telecommunications
National & Kapodistrian University of Athens, Greece, and
ATHENA Research & Innovation Center, Greece
emiris@di.uoa.gr
https://orcid.org/0000-0002-2339-5303

Vissarion Fisikopoulos
Oracle, Greece
vissarion.fysikopoulos@oracle.com
https://orcid.org/0000-0002-0780-666X

Abstract

We examine volume computation of general-dimensional polytopes and more general convex bodies, defined as the intersection of a simplex by a family of parallel hyperplanes, and another family of parallel hyperplanes or a family of concentric ellipsoids. Such convex bodies appear in modeling and predicting financial crises. The impact of crises on the economy (labor, income, etc.) makes its detection of prime interest for the public in general and for policy makers in particular. Certain features of dependencies in the markets clearly identify times of turmoil. We describe the relationship between asset characteristics by means of a copula; each characteristic is either a linear or quadratic form of the portfolio components, hence the copula can be constructed by computing volumes of convex bodies.

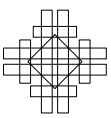
We design and implement practical algorithms in the exact and approximate setting, we experimentally juxtapose them and study the tradeoff of exactness and accuracy for speed. We analyze the following methods in order of increasing generality: rejection sampling relying on uniformly sampling the simplex, which is the fastest approach, but inaccurate for small volumes; exact formulae based on the computation of integrals of probability distribution functions, which are the method of choice for intersections with a single hyperplane; an optimized Lawrence sign decomposition method, since the polytopes at hand are shown to be simple with additional structure; Markov chain Monte Carlo algorithms using random walks based on the hit-and-run paradigm generalized to nonlinear convex bodies and relying on new methods for computing a ball enclosed in the given body, such as a second-order cone program; the latter is experimentally

* The views expressed are those of the authors and do not necessarily reflect official positions of the European Commission.

[1] Calès acknowledges the financial support of the European Commission through its Proof-of-Concept program.

[2] Chalkis and Emiris are partially supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 734242 (Project LAMBDA).

34th International Symposium on Computational Geometry (SoCG 2018).
Editors: Bettina Speckmann and Csaba D. Tóth; Article No. 19; pp. 19:1–19:15



Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

extended to non-convex bodies with very encouraging results. Our C++ software, based on CGAL and Eigen and available on `github`, is shown to be very effective in up to 100 dimensions. Our results offer novel, effective means of computing portfolio dependencies and an indicator of financial crises, which is shown to correctly identify past crises.

2012 ACM Subject Classification Theory of computation → Computational geometry, Theory of computation → Random walks and Markov chains

Keywords and phrases Polytope volume, convex body, simplex, sampling, financial portfolio

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.19

Related Version A full version of this paper is available at `https://arxiv.org/abs/1803.05861`

1 Introduction

1.1 Financial context and motivation

Modern finance has been pioneered by Markowitz who set a framework to study choice in portfolio allocation under uncertainty [29], and for which he was awarded the Nobel Prize in economics (1990). Markowitz characterized portfolios by their return and their risk which is defined as the variance of the portfolios' returns. An investor would build a portfolio that maximizes its expected return for a chosen level of risk. In the same way, by choosing a level of expected return, an investor may construct a portfolio which minimizes risk. It has since become common for asset managers to optimize their portfolio within this framework. And it has led a large part of the empirical finance research to focus on the so-called efficient frontier which is defined as the set of portfolios presenting the lowest risk for a given expected return. Figure 1 (left panel) presents such an efficient frontier. The region on the left of the efficient frontier represent the portfolios domain.

Interestingly, despite the fact that this framework considers the whole set of portfolios, no attention has been given to the distribution of portfolios. Figure 1 (middle panel) presents such distribution where 10.000.000 portfolios have been sampled as presented later in Section 2.1. When comparing the contour of the empirical portfolios distribution, i.e. the region over which at least one random portfolio lies, and the portfolio domain bounded by the efficient frontier in Figure 1 (right panel), we observe that the density of portfolios along the efficient frontier is dim and that most of the portfolios are located in a small region.

We also know from the financial literature that financial markets exhibit 3 types of behavior. In normal times, stocks are characterized by slightly positive returns and a moderate volatility, in up-market times (typically bubbles) by high returns and low volatility, and during financial crises by strongly negative returns and high volatility, see e.g. [5] for details. So, following Markowitz' framework, in normal and up-market times, the stocks and portfolios with the lowest volatility should present the lowest returns, whereas during crises those with the lowest volatility should present the highest returns. These features, also called "stylized facts" in the financial literature, motivate us to describe the time-varying dependency between portfolios' returns and volatility.

However this dependency is difficult to capture from the usual mean-variance representation, as in Figure 1 (middle panel), so we will rely on the copula representation of the portfolios distribution. A copula is a bivariate probability distribution for which the marginal probability distribution of each variable is uniform. As we following Markowitz' framework,

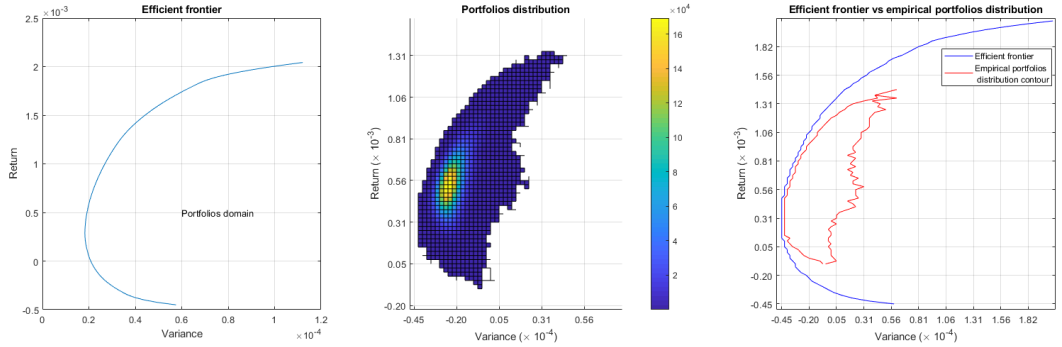


Figure 1 (left) Efficient frontier; (middle) Empirical portfolio distribution by portfolios' return and variance; (right) Efficient frontier in blue and contour of the empirical portfolio distribution in red. The market considered is made of the 19 sectoral indices of the DJSTOXX 600 Europe. The data is from October 16, 2017 to January 10, 2018.

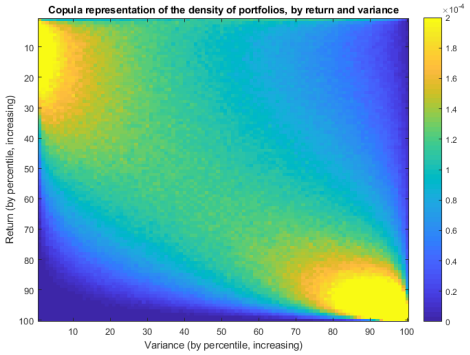


Figure 2 Copula representation of the portfolios distribution, by return and variance. The market considered is made of the 19 sectoral indices of the DJSTOXX 600 Europe. The data is from October 16, 2017 to January 10, 2018.

the variables considered are the portfolios' return and variance. Figure 2 illustrates such a copula and shows a positive dependency between portfolios returns and variances. Each line and column sum to 1% of the portfolios.

The methods introduced here can be used to study other dependencies such as the momentum effect [23] which is implied by the dependencies of asset returns with their past returns.

The dependencies mentioned here are important because

- through the return/volatility dependency, the detection of crisis raises policy makers awareness and allows them to act accordingly with potentially large implications in citizens' life (employment, wages, pensions, etc).
- the momentum, if persistent, questions the efficiency of financial markets, a strong assumption which still cannot be proven wrong.

Interestingly, the copulas can be computed over a single period of time making the information available as early as the sample allows. The copula for the momentum dependency can be computed over very short periods (even intra-daily). The copula for the return/volatility dependency requires the estimation of the stock returns variance-covariance matrix which

has to be estimated over a sufficiently large period of time to be reliable thus delaying the detection of crisis.[3]

In the general case, the framework to describe the dependencies is as follows. First, as the set of portfolios, we consider the canonical d-dimensional simplex $\Delta^d \subset \mathbb{R}^{d+1}$ where each point represents a portfolio and $d+1$ is the number of assets. The vertices represent portfolios composed entirely of a single asset. The portfolio weights, i.e. fraction of investment to a specific asset, are non-negative and sum to 1. This is the most common investment set in practice today, as portfolio managers are typically forbidden from short-selling or leveraging. Second, considering some asset characteristic ac quantified by $C \in \mathbb{R}^{d+1}$, we define a corresponding quantity $f_{ac}(\omega, C)$ for any portfolio $\omega \in \Delta^d$. For instance, considering the vector of asset returns $R \in \mathbb{R}^{d+1}$, ω has the return $f_{ret}(\omega, R) = R^T\omega$. Then, we define the cross-sectional score of a given portfolio ω^* as

$$\rho_{ac} = \frac{\text{vol}(\Delta^*)}{\text{vol}(\Delta^d)}, \quad \text{where } \Delta^* = \{\omega \in \Delta^d \,:\, f(\omega, C) \leq f(\omega^*, C)\},$$

which corresponds to the share of portfolios with a return lower or equal to $R^* = R^T\omega^*$. This score corresponds to the cumulative distribution function of $f_{ac}(\omega, C)$ where the portfolios are uniformly distributed over the simplex. In the following, we consider the cases where f_{ac} is a linear combination or a quadratic form of C. Finally, the relationship between two asset characteristics ac_1 and ac_2 is presented in the form of a *copula* whose marginals are ρ_{ac_1}, ρ_{ac_2}. In our applications, the asset characteristics considered are the assets' returns and variances, and their values correspond to a linear combination of the returns and a quadratic form of the returns, respectively.

A copula is computed by slicing a simplex, i.e. the set of portfolios, along the asset characteristics. Thus, these questions are formulated in terms of convex bodies defined by intersecting simplices on one hand by a family of parallel hyperplanes and, on the other hand, by another family of parallel hyperplanes in the linear case or a family of concentric ellipsoids in the quadratic case. The latter case yields non-convex bodies between two ellipsoids.

1.2 Previous work

The cross-sectional score of portfolio returns has been introduced in [32] and it is estimated by means of a quasi-Monte Carlo method. The applications have been limited in terms of dimensions: the 30 DAX components and the 24 MSCI Netherlands components in [32], the 35 components of the IBEX in [33]. This score has also been proposed in [4], for the set of long/short equally weighted zero-dollar portfolios and whose estimation relying on combinatorics and statistics is computationally limited to around 20 dimensions, and in [3] where the focus was not on a precise score.

Given that volume computation of polytopes is #P-hard for both V- and H-representations [17] and no poly-time algorithm can achieve better than exponential error [18], the problem is not expected to admit of an efficient deterministic algorithm in general dimension. Developing algorithms for volume computation has received a lot of attention in the exact setting [7]. In the approximate setting, following the breakthrough polynomial-time algorithm by random walks [16], several algorithmic improvements ensued. The current best theoretical bounds are in [26] and for polytope sampling in [27]. Interestingly, only two pieces of software offer

[3] Methods exist to estimate the stock returns variance-covariance matrix over short periods, see e.g. the range-based estimation method [5]. However they usually requires high-frequency data and are not widely used. These methods are beyond the scope of this paper.

practical algorithms in high dimension: `VolEsti`, a public-domain C++ implementation that scales to a few hundred dimensions [19], based on the Hit-and-Run paradigm [28], and the Matlab implementation of [10], which seems competitive to `VolEsti` in very high dimensions. Sampling from non-convex bodies appears in experimental works, with very few methods offering theoretical guarantees, e.g. in star shaped bodies [9] or, more recently, in [1].

1.3 Our contribution

We design and implement the following different approaches for volume computation: Efficient sampling from the simplex and using rejection to approximate the target volume, which is fast but inaccurate for small volumes. Exact formulae of integrals of appropriate probability distribution functions, which are implemented for the case of a single hyperplane. Optimizing the use of Lawrence's sign decomposition method, since the polytopes at hand are shown to be simple with extra structure; a major issue here is numerical instability. Extending state-of-the-art random walks, based on the hit-and-run paradigm, to convex bodies defined as the intersection of linear halfspaces and ellipsoids. The latter is experimentally generalized to non-convex bodies defined by two ellipsoids with same quadratic form, and accurate approximations are obtained under certain mild conditions.

Our randomized algorithms for volume approximation extend `VolEsti`, where the main problem to address is to compute the maximum inscribed ball of the convex body P a.k.a. Chebychev ball. This reduces to a linear program when P is a polytope. For a convex body defined by intersecting a polytope with k balls, the question becomes a second-order cone program (SOCP) with k cones. When interchanging input balls with ellipsoids, the SOCP yields a sufficiently good approximation of the Chebychev ball.

Our implementations are in C++, lie in the public domain (`github`), are based on CGAL, rely on Eigen for linear algebra, on `Boost` for random number generators, and experiment with two SOCP solvers for initializing random walks. Our software tools are general and of independent interest. They are applied to allow us to extend the computation of a portfolio score to up to 100 dimensions, thus doubling the size of assets studied in financial research. We thus provide a new description of asset characteristics dependencies. Our methods allow us to propose and to effectively compute a new indicator of financial crises, which is shown to correctly identify all past crises with which we experimented. More importantly, it allows us to establish that periods of momentum nearly never overlap with the crisis events, which is a new result in finance.

The rest of the paper is organized as follows. The next section presents the convex bodies that arise from our financial modeling; we overview methods for representing and uniformly sampling from the simplex. Section 3 considers volumes defined as the intersection of a simplex and one hyperplane or more hyperplanes, the latter being organized in at most two families of parallel hyperplanes. Section 4 studies convex and non-convex bodies defined as the intersection of a simplex and an ellipsoid. Implementations are discussed in Section 5, along with experiments. We conclude with current work and open questions. Figures, proofs and tables of experiments that do not fit here are included in the full version of the paper in [8].

2 Convex bodies and Financial modeling

We analyze real data consisting of regular interval (e.g. daily) returns of assets such as the constituents of the Dow Jones Stoxx 600 Europe™ (DJ600). These are points in real space of dimension $d = 600$, respectively: $r_i = (r_{i,1}, \ldots, r_{i,d}) \in \mathbb{R}^d$, $i \geq 1$.

We apply the methodology to a subset of assets drawn from the DJ 600 constituents[4]. Since not all stocks are tracked for the full period of time, we select the 100 assets with the longest history in the index[5], and juxtapose:

- stock returns and stock returns covariance matrix over the same period to detect crises,
- stock returns and past stock returns to observe any momentum effect,

In financial applications, one considers compound returns over periods of k observations, where typically $k = 20$ or $k = 60$; the latter corresponds to roughly 3 months when observations are daily. Compound returns are obtained using k observations starting at the i-th one where the j-th coordinate corresponds to asset j and the component j of the new vector equals:

$$(1 + r_{i,j})(1 + r_{i+1,j}) \cdots (1 + r_{i+k-1,j}) - 1, \quad j = 1, \ldots, d.$$

This defines the normal vector to a family of parallel hyperplanes, whose equations are fully defined by selecting appropriate constants. The second family of parallel hyperplanes is defined similarly by using an adjacent period of k observations.

The covariance matrix of the stock returns is computed using the shrinkage estimator of [25],[6] as it provides a robust estimate even when the sample size is short with respect to the number of assets. A covariance matrix C defines a family of ellipsoids centered at the origin $0 \in \mathbb{R}^d$ whose equations $x^T Cx = c$ are fully specified by selecting appropriate constants c.

To compute the copulas, we determine constants defining hyperplanes and ellipsoids so that the volume between two consecutive such objects is 1% of the simplex volume. The former are determined by bisection using the Varsi's exact formula. For ellipsoids $E(x) = c_i$, we look for the c_i's by sampling the simplex, then evaluating $E(x)$ at each point. The values are sorted and the c_i selected so as to define intervals containing 1% of the values. Two consecutive ellipsoids intersecting the simplex and the family of parallel hyperplanes define a non-convex body for which we practically extend `VolEsti` algorithm.

The volume between two consecutive hyperplanes and two consecutive ellipsoids defines the density of portfolios whose returns and volatilities lie between the specified constants. We thus get a copula representing the distribution of the portfolios with respect to the portfolios returns and volatilities.

The main problem is to compute all the volumes that arise from the intersection of the two families with the unit simplex. We have to handle three types of full dimensional bodies and thus we develop or use existing methods for three different problems. The first is to compute the volume of the polytope defined by the intersection of the unit simplex with four hyperplanes which are pairwise parallel. The second arises when an ellispoid intersects the unit simplex and a family of parallel hyperplanes. The third is to compute the volume of a non-convex body defined by the intersection of two concentric ellipsoids with a simplex and a family of parallel hyperplanes.

We develop and use four methods in total. The first (M1) is an exact formula for the volume defined by the intersection of simplex with a hyperplane. The second (M2 or s/r) is to sample the unit simplex and approximate all the volumes directly. The third method (M3) is the optimized Lawrence formula for simple polytopes and is used for the first problem.

---

[4] The data used is from Bloomberg™.. It is daily and ranges from 01/01/1990 to 31/11/2017.

[5] This implies a survivor bias, but we use it to assess the effectiveness of the methodology. One would wish to keep 600 constituents, replacing the exiting stocks with the entering ones along the sample.

[6] Matlab code on `http://www.econ.uzh.ch/en/people/faculty/wolf/publications.html`.

The fourth method (M4) is the generalization of the `VolEsti` algorithm to non-linear and non-convex bodies.

2.1 Simplex representation and sampling

This subsection sets the notation, surveys methods for uniform sampling of the simplex, and discusses their efficient implementation.

The d-dimensional simplex $\Delta^d \subset \mathbb{R}^{d+1}$ may be represented by barycentric coordinates $\lambda = (\lambda_0, \ldots, \lambda_d)$ s.t. $\sum_{i=0}^{d} \lambda_i = 1$, $\lambda_i \geq 0$. The points are $\sum_{i=0}^{d} \lambda_i v_i$, where $v_0, \ldots, v_d \in \mathbb{R}^d$ are affinely independent. It is convenient to use a full-dimensional simplex, by switching to Cartesian coordinates $x = (x_1, \ldots, x_d)$ using transformation $m_{bc} : \mathbb{R}^{d+1} \mapsto \mathbb{R}^d : \lambda \to x = M(\lambda_1, \ldots, \lambda_n)^T + v_0$, where $M = [v_1 - v_0 \cdots v_d - v_0]$, is a $d \times d$ invertible matrix. The inverse transform is:

$$m_{cb} : \mathbb{R}^d \mapsto \mathbb{R}^{d+1} : x \to \lambda = \begin{bmatrix} -1_d^T \\ I_d \end{bmatrix} M^{-1}(x - v_0) + \begin{bmatrix} 1 \\ 0_d \end{bmatrix}, \tag{1}$$

where $0_d, 1_d$ are d-dimensional column vectors of 1's and 0's, respectively, and I_d is the d-dimensional identity matrix.

A number of algorithms exist for sampling, where some have been rediscovered, while others contain errors; see the survey [37]. Let us start with a unit simplex in Cartesian coordinates. A $O(d \log d)$ algorithm is the following [12, 13, 34]: Generate d distinct integers uniformly in $\{1, \ldots, K-1\}$, where K is the largest representable integer. Sort them as follows: $x_0 = 0 < x_1 < \cdots < x_{d+1} = K$. Now $(x_i - x_{i-1})/K$, $i = 1, \ldots, d$, defines a uniform point. Assuming we possess a perfect hash-function, the choice of distinct integers takes $O(d)$. For $d > 60$ we implement a variant of Bloom filter to guarantee distinctness.

A linear-time algorithm is given in [35], which is is generally the algorithm of choice, although it is slower for $d < 80$: (1) Generate $d+1$ independent unit-exponential random variables y_i by uniformly sampling real value $x_i \in (0, 1)$ and setting $y_i = -\log x_i$, (2) Normalize the y_i's by their sum $s = \sum_{i=0}^{d} y_i$, thus obtaining a uniformly distributed point $(y_0/s, \ldots, y_d/s)$ on the d-dimensional canonical simplex lying in $\mathbb{R}^{d+1}$, (3) Project this point along the x_0-axis to $(y_1/s, \ldots, y_d/s)$, which is a uniform point in the full-dimensional unit simplex.

To sample an arbitrary simplex, we can map sampled points from the unit simplex by transformation (1), which preserves uniformity. Due to applying the transformation, the complexity is $O(d^2)$ to generate a uniform point. The same complexity, though slower in practice, is achieved in [22].

Sampling could be used in to approximate all the volumes that arise when two families of parallel hyperplanes intersect with a simplex. One can sample the simplex and count the percentage of points in the region of interest (we call this method M2 or s/r). The complexity is $O(kd)$ to generate k points. In the case of a family of ℓ parallel hyperplanes, all sample points are evaluated at the hyperplane linear polynomials in time $O(kd)$. Given the ℓ constant terms characterizing the hyperplanes, for each point we perform a binary search so as to decide in which layer it lies. Hence the total complexity is $O(k \log \ell)$, which is dominated since $\ell \leq 100$ typically. Given a family of ℓ ellipsoids with same quadratic form intersecting a simplex, the method requires $O(kd^2)$ to evaluate all sample points and $O(k \log \ell)$ to assign them to layers.

3 Intersection with hyperplanes

This section considers computing the volume of the intersection of a simplex and one or more linear halfspaces. The most general case is to be given two families of parallel hyperplanes and consider all created polytopes. We assume that the simplex is given in V-representation, i.e. as a set of vertices, and the hyperplanes by their equations.

We can always transform the simplex to be a unit full-dimensional simplex with the origin as one vertex by the transformation of Section 2.1. The same transform applies to the hyperplanes, and volume ratios as preserved.

Surprisingly, there exist an exact, iterative formula for the volume defined by intersecting a simplex with a hyperplane. A geometric proof is given in [38], by subdividing the polytope into pyramids and, recursively, to simplices. We implement a somewhat simpler formula [2], which also requires $O(d^2)$ operations. Let $H = \{(x_1, \ldots, x_d) | \sum_{i=1}^{d} a_i x_i \leq z\}$ be the linear halfspace.

1. Compute $u_j = a_j - z$, $j = 1, \ldots, d$. Label the nonnegative u_j as $Y_1, \ldots, Y_K$ and the negatives as $X_1, \ldots, X_J$. Initialize $A_0 = 1, A_1 = A_2 = \cdots = A_K = 0$.
2. For $h = 1, 2, \ldots, J$ repeat: $A_k \longleftarrow \dfrac{Y_k A_k - X_h A_{k-1}}{Y_k - X_h}$, for $k = 1, 2, \ldots, K$.

If $\Delta^d \subset \mathbb{R}^d$ is the unit simplex then, for $h = J$, $A_K = \mathrm{vol}(\Delta^d \cap H)/\mathrm{vol}(\Delta^d)$.

We now consider simple polytopes defined by a constant number of families of parallel hyperplanes; In our application there are two such families. The defined polytopes are simple, i.e., all vertices are defined at the intersection of d hyperplanes, assuming that no hyperplane contains any of the simplex vertices and, moreover, two hyperplanes do not intersect on a simplex edge at the same point.

For a simple polytope P, the decomposition by Lawrence [24] picks $c \in \mathbb{R}^d$, $q \in \mathbb{R}$ such that $c^T x + q$ is not constant along any edge, i.e. $c, -c$ do not lie on the normal fan of any edge. For each vertex v, let $A(v)$ be the $d \times d$ matrix whose columns correspond to the equations of hyperplanes through v. Then $A(v)$ is invertible and vector $\gamma(v)$ such that $A(v)\gamma(v) = c$ is well defined up to a permutation. The assumption on c assures no entry vanishes, then

$$\mathrm{vol}(P) = \frac{1}{d!} \sum_{v} \frac{(c^T v + q)^d}{|\det A(v)| \prod_{i=1}^{d} \gamma(v)_i}.$$

The computational complexity is $O(d^3 n)$, where n is the number of vertices. We set $q = 0$ for simplicity in the implementation. The main drawback of Lawrence's decomposition remains numerical instability when executed with floating point numbers, and high bit complexity, when executed over rational arithmetic. The latter is indispensable for $d > 30$ in our applications, because then numerical results become very unstable.

To compute the volume defined by the intersection of a simplex and two arbitrary hyperplanes, we exploit the fact that the simplex is unit in order to compute more effectively the determinants and the solutions of the linear system. The hardest case is when vertex v is defined by the two arbitrary hyperplanes H_a, H_b, the supporting hyperplane $H_0 : \sum_{i=1}^{d} x_i = 1$, and $d-3$ hyperplanes of the form $H_i : x_i = 0$. Then we could compute $\gamma(v)_i$, $i = 1, \ldots, d$ by solving the linear system in $O(d)$. The corresponding determinant is computed in $O(1)$. For the number of vertices we show the Lemma below.

▶ **Lemma 1.** *Polytopes in H-representation, defined by intersecting the simplex with two arbitrary hyperplanes in $\mathbb{R}^d$, have $O(d^2)$ vertices, which are computed in $O(1)$ each.*

The proof of lemma 1 is given in [8]. Lawrence's formula requires both H- and V-representation. In our setting, the H-representation is known, but the previous lemma allows us to obtain vertices as well.

▶ **Proposition 2.** *Let us consider polytopes defined by intersecting the simplex with two arbitrary hyperplanes. The total complexity of the Lawrence sign decomposition method, assuming that the H-representation is given, is* $O(d^3)$.

The entire discussion extends to polytopes defined by two families of parallel hyperplanes. The matrices $A(v)$ remain of the same form because each vertex is incident to at most one hyperplane from each family.

4 Intersection with ellipsoids

This section considers more general convex bodies, defined as a finite, bounded intersection of linear and nonlinear halfspaces. For this, we extend the polynomial-time approximation algorithm in `VolEsti` [19] so as to handle nonlinear constraints. Our primary motivation here is computing the volume of the intersection of a simplex with an ellipsoid in general dimension.

4.1 Random walks

The method in [19] follows the Hit-and-Run algorithm in [28], and is based on an approximation algorithm in $O^*(d^5)$. It scales in a few hundred dimensions by integrating certain algorithmic improvements to the original method. We have to generalize the method because the input is not a polytope but a general convex body, while `VolEsti` works for d-polytopes. It suffices to solve two subproblems: Compute the maximum inscribed ball of the convex body a.k.a. Chebychev ball, and compute the intersection points of a line that crosses the interior of the convex body P with the boundary of P.

The first problem is treated in the next subsection. For the second one, when the body is the intersection of linear and quadratic halfspaces, it suffices to solve systems of linear or quadratic equations. In our case where P has few input hyperplanes we can optimize that procedure by transforming a base of our polytope to an orthonormal base thus obtaining very simple linear systems. Formally, every ray ℓ in Coordinate Direction Hit-and-Run is of the form $p + \lambda e_k$ and parallel to $d-1$ simplex facets. The roots of $\lambda^2 + 2\lambda p_k + |p|^2 - R^2$ define the intersection of a sphere with radius R, centered at the origin, and a coordinate direction ray ℓ. If C is the matrix of an ellipsoid centered at the origin its intersections with ℓ are roots of $C_{kk}x^2 + bx + c = 0$, where $b = 2C_{kk}p_k + 2\sum_{j=k+1}^{d} C_{kj}p_j + 2\sum_{i=0}^{k-1} C_{ik}p_i$ and $c = \sum_{i=0}^{d} C_{ii}p_i^2 + 2\sum_{j=i+1}^{d} C_{ij}p_ip_j$. Computing the roots, and keeping the largest negative and smallest positive λ is quite fast.

In our application, there are non-convex bodies defined by the intersection of two parallel hyperplanes, two concentric ellipsoids and a simplex. We thus modify `VolEsti` in order to compute the non convex volume. We make two major changes. First, in ray shooting, we have to check whether one quadratic equation has only complex solutions, which implies the ray does not intersect the ellipsoid. For λ, we take the largest negative and the smallest positive root in every step as well. Second, for the initial interior point, we sample from the unit simplex and when we find a point inside the intersection we stop and use it for initialization. We define an inscribed ball with this center and radius equal to some small $\epsilon > 0$. We stop the algorithm when we find the first inscribed ball as described in the next subsection. So we can set ϵ sufficiently small so it always defines an inscribed ball in practice, but the enclosing ball is enough to run the algorithm and do not stop until we find an inscribed ball.

The method works fine for $d < 35$ using the same walk length and number of points as for the convex case, and has time complexity and accuracy competitive to running `VolEsti` on the convex set defined by one ellipsoid. For $d > 35$, the method fails to approximate volume for most of the cases. This should be due to inaccurate rounding bodies and the inscribed ball we define. Given these first positive results, various improvements are planned.

4.2 Chebychev ball

This section offers methods for computing a ball inside the given convex region. Ideally, this is the largest inscribed ball, aka Chebychev ball, but a smaller ball may suffice. Computing the Chebychev ball reduces to a linear program when P is a polytope (p. 148 in [6]). For general convex regions, more general methods are proposed.

At the very least, one point must be obtained inside the convex region. When we do not have the Chebychev ball, an issue is that concentric balls with largest radii will again be entirely contained in the convex region, thus wasting time in the computation. In practice we use the one interior point as center of an enclosing ball, then reduce the radius until the first inscribed ball. To decide whether a given ball is inscribed, with high probability, we check whether all boundary points in Hit-and-Run belong to the sphere instead of any other constraint.

For a convex body that comes from intersecting a polytope with k balls the problem becomes a Second-Order Cone Program (SOCP) with k cones. However in our case we need to consider input ellipsoids. Assume that we transformed the ellipsoid to a ball $B' = \{x'_c + u' \ : \ \|u'\| \leq r'\}$, and applied the same transformation to the simplex to have $a_i x \leq b_i$ for $i \in [d+1], a_i \in \mathbb{R}^d, b_i \in \mathbb{R}$. The following SOCP computes the maximum ball $B = \{x_c + u \ : \ \|u\| \leq r\}$ in the intersection of the simplex and B':

$$\max \ r, \quad \text{subject to} : a_i^T x_c + r||a_i|| \leq b_i, \ ||x'_c - x_c|| \leq r' - r.$$

There are several ways to solve SOCP's such as to reformulate it to as a semidefinite program or perform a quadratic program relaxation. Moreover, since in our case we only have a single cone we could utilize special methods as in [20]. However, for our case it suffices to use the generic SOCP solver from [14] as it is very efficient; for a random simplex and ball, it takes 0.06 sec in $d = 100$ and < 20 sec in $d = 1000$, on Matlab using `ecos` and `yalmip` packages.

It is possible to apply the inverse transformation and get an inscribed ellipsoid, which is not necessarily largest possible. However we can use the maximum inscribed ball in that ellipsoid as an approximation of the Chebychev ball, by taking the center of that ellipsoid and the minimum eigenvalue of its matrix as the radius.

4.3 Portfolios' variances expressed by ellipsoids

In our financial application, portfolios are points in the unit d-dimensional simplex $\Delta^d \subset \mathbb{R}^{d+1}$ defined as the convex hull of $v_0, \ldots, v_d \in \mathbb{R}^d$, where v_i lies on the i-th axis. The simplex lies in hyperplane $\sum_{i=0}^{d} \lambda_i = 1$. To model levels of portfolios' variances, a family of full-dimensional ellipsoids in $\mathbb{R}^{d+1}$, centered at the origin, is defined by the covariance matrix C of asset returns. We wish to compute the volume of intersections of this family with the simplex and, moreover, with a family of hyperplanes on the simplex. Rejection sampling would work in this context, however methods employing random walks require a full-dimensional convex body. Given a full $(d+1)$-dimensional ellipsoid $G : \lambda^T C \lambda - c \leq 0$ centered at the origin, where $C \in \mathbb{R}^{(d+1)\times(d+1)}$ is symmetric positive-definite, we compute the equation of

the ellipsoid defined $G \cap \Delta^d \subset \mathbb{R}^d$, by imposing the constraint $\sum_{i=0}^{d} \lambda_i = 1$ by transform m_{cb} in expression (1), thus obtaining:

$$(x - v_0)^T \left(M^{-T} \left[-1 \, I_d \right] C \begin{bmatrix} 1 \\ 0_d \end{bmatrix} M^{-1} \right) (x - v_0) + A(x - v_0) = c',$$

where the expression in parenthesis is the matrix defining the new d-dimensional ellipsoid in Cartesian coordinates, and $A \in \mathbb{R}^{d \times d}, c' \in \mathbb{R}$ are obtained by direct calculation. Similarly the simplex maps to Cartesian coordinates.

5 Implementation and experiments

5.1 Implementation

Our implementations are in C++, lie in the public domain[7], and are using CGAL and Eigen. All experiments of the paper have been performed on a personal computer with Intel Pentium G4400 3.30GHz CPU and 16GB RAM. Times are averaged over 100 runs. All the details of the experiments and the corresponding Tables are given in the full version of the paper in [8].

We test the following convex bodies: a d-simplex intersected with: (1) two arbitrary halfspaces, (2) two parallel halfspaces, (3) an ellipsoid, (4) two parallel halfspaces and two cocentric ellipsoids (non convex body).

In general, M1 is preferred when available. Method M2 is the fastest and scales easily to 100 dimensions, so it is expected to be useful for larger dimensions. However, for small volumes its accuracy degrades; sampling more points makes it slower than M4. The latter is thus the method of choice for volumes $< 1\%$ of the simplex volume, but it is not clear whether it would be fast beyond $d = 100$. Method M3 is useful, even for small volumes, but it cannot scale to $d = 100$ due to numerical instability; if we opt for exact computing, it becomes too slow.

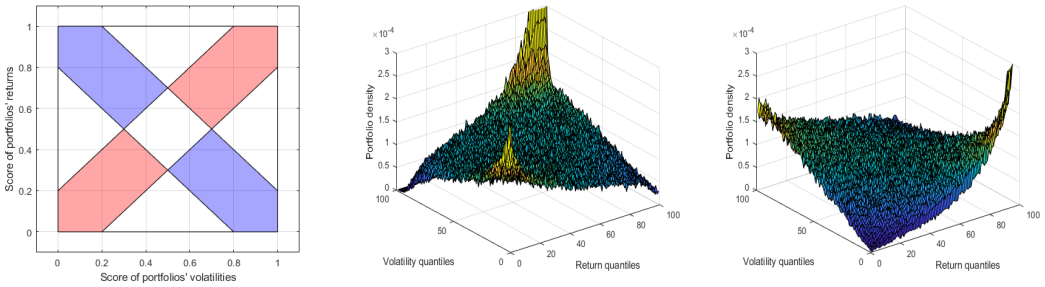
5.2 Financial modeling with real data

When working with real data and in order to build the indicator, we wish to compare the densities of portfolios along the two diagonals. In normal and up-market times, the portfolios with the lowest volatility present the lowest returns and the mass of portfolios should be on the up-diagonal. During crisis the portfolios with the lowest volatility present the highest returns and the mass of portfolios should be on the down-diagonal. Thus, defining up- and down-diagonal bands, we construct the indicator as the ratio of the mass on down-diagonal band over the one on the up-diagonal band, discarding the intersection of the two. Figure 3 illustrates the way the indicator is built.

In the following, the indicator is computed using copulas estimated using the sampling method, drawing 500,000 points. Computing the indicator over a rolling window of $k = 60$ days and with a band of $\pm 10\%$ with respect to the diagonal, we report in Table 1 all the periods over which the indicator is greater than 1 for more than 60 days. The periods should be more than 60 days to avoid the detection of isolated events whose persistence is only due to the auto-correlation implied by the rolling window.

We compare these results with the database for financial crises in European countries proposed in [15], and the longest periods found with the indicator coincide to crisis. The first

[7] https://github.com/TolisChal/volume_approximation



(a) Diagonal bands considered to build the indicator. **(b)** Returns/variance relationship on the 1^{st} September 1999. **(c)** Returns/variance relationship on the 1^{st} September 2000.

Figure 3 Illustration of the indicator. Panel (a): diagonal bands. Panels (b) and (c): copulas obtained during the dot-com bubble and at the beginning of the bubble burst, respectively. Blue= low density of portfolios, yellow=high density of portfolios.

crisis (from May 1990 to Dec. 1990) corresponds to the early 90's recession, the second one (from May 2000 to May 2001) to the dot-com bubble burst, the third one (from Oct. 2001 to Apr. 2002) to the stock market downturn of 2002, and the fourth one (from Dec. 2007 to Aug. 2008) to the sub-prime crisis. The remaining periods are shorter and correspond to periods of financial turmoil. They were not officially recognized as financial crisis.

Regarding the momentum effect, i.e. the effect of the compound returns of the last 60 days on the following 60-day compound returns. We observe that there were only 10 events of lasting momentum effect, mostly around the 1998-2004 period. We remark that they nearly never overlap with the crisis events, with the exception of the end of 2011. To the authors' knowledge, this last result is new in finance. Indeed, following the recommendations in [23], most of the literature focuses on the coincidence of momentum periods with economic crisis with inconclusive results, see e.g. [21] and [36]. By contrast, very few focus on its coincidence with financial crisis, probably because of the lack of financial crisis dating. Notably [11] find that momentum crash periods occur after the market has fallen, and when volatility is high and the market is recovering, i.e. at the end of a financial crisis. These two results are consistent.

6 Conclusion and future work

Since runtimes are very reasonable, we plan to extend our study to larger subsets of assets of DJ 600 and eventually the whole index in $d = 600$. Another extension is to consider polytopes defined by intersections of both families of parallel hyperplanes and the family of ellipsoids, thus creating 3-D diagrams of dependencies, which have never been studied in finance: one difficulty is to model the outcome since visualization becomes intricate.

An obvious enhancement is to parallelize our algorithms, which seems straightforward. Some other challenges are to obtain theoretical guarantees for the sampling-rejection methods and to extend the volume formula to the intersection with a ellipsoid. In [31], they propose a method to approximate the distribution f of quadratic forms in gamma random variables which is a similar problem to that in [30] (Section 3). It consists in fitting f with a generalized gamma distribution by matching its first 3 moments with those of f and to adjust the distribution with a polynomial in order to fit the higher moments. To get an approximation with a polynomial of degree k, the method requires the first $2k$ moments.

Table 1 Phases of crisis (a) and momentum effect (b) detected with the indicator.

Start date	End date	Duration (days)
02-May-1990	20-Dec-1990	166
06-May-1992	14-Aug-1992	72
06-Oct-1994	27-Jan-1995	80
08-Apr-1996	24-Jul-1996	77
01-Jul-1997	13-Oct-1997	74
03-Mar-1999	01-Jun-1999	61
04-May-2000	09-May-2001	258
05-Oct-2001	05-Apr-2002	124
25-Feb-2004	28-May-2004	65
18-Nov-2005	11-Apr-2006	101
20-Dec-2007	04-Aug-2008	157
28-Dec-2010	12-Apr-2011	75
18-Oct-2011	16-Jan-2012	63
08-Oct-2013	04-Feb-2014	82
04-Jun-2015	05-Oct-2015	87
30-Nov-2015	03-Mar-2016	66

(a) All periods over which the return/volatility indicator is greater than one for more than 60 days.

Start date	End date	Duration (days)
26-Dec-1990	16-Apr-1991	79
18-Oct-1993	11-Jan-1994	61
11-Aug-1998	24-Nov-1998	75
08-Nov-1999	04-Apr-2000	105
22-May-2001	04-Sep-2001	75
14-Jun-2002	09-Oct-2002	83
18-Oct-2002	27-Mar-2003	111
20-Aug-2004	21-Dec-2004	87
13-Oct-2006	19-Jan-2007	67
26-Jul-2011	21-Dec-2011	106

(b) All periods over which the momentum indicator is greater than one for more than 60 days.

In the case of a quadratic form in d random variables, the moment of order m is obtained by a sum over all the partitions of m into d^2 terms. The number of partitions makes the computation of moments challenging even for $d \geq 5$.

References

1 Y. Abbasi-Yadkori, P.L. Bartlett, V. Gabillon, and A. Malek. Hit-and-run for sampling and planning in non-convex spaces. In *Proc. 20th Intern. Conf. Artificial Intelligence & Stat. (AISTATS)*, pages 888–895, 2017. URL: `http://proceedings.mlr.press/v54/abbasi-yadkori17a.html`.

2 M. Maswood Ali. Content of the frustum oa a simplex. *Pacific J. Math.*, 48(2):313–322, 1973.

3 A. Banerjee and C-H. Hung. Informed momentum trading versus uninformed “naive" investors strategies. *J. Banking & Finance*, 35(11):3077–3089, 2011.

4 M. Billio, L. Calès, and D. Guéguan. A cross-sectional score for the relative performance of an allocation. *Intern. Review Appl. Financial Issues & Economics*, 3(4):700–710, 2011.

5 M. Billio, M. Getmansky, and L. Pelizzon. Dynamic risk exposures in hedge funds. *Comput. Stat. & Data Analysis*, 56(11):3517–3532, 2012. `doi:10.1016/j.csda.2010.08.015`.

6 S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, UK, 2004.

7 B. Büeler, A. Enge, and K. Fukuda. Exact volume computation for polytopes: A practical study. In G. Kalai and G.M. Ziegler, editors, *Polytopes: Combinatorics and Computation*, volume 29 of *Math. & Statistics*, pages 131–154. Birkhäuser, Basel, 2000.

8 L. Calès, A. Chalkis, I.Z. Emiris, and V. Fisikopoulos. Practical volume computation of structured convex bodies, and an application to modeling portfolio dependencies and

financial crises. *CoRR*, arXiv:1803.05861, March 2018. URL: https://arxiv.org/abs/1803.05861.

9 K. Chandrasekaran, D. Dadush, and S. Vempala. Thin partitions: Isoperimetric inequalities and sampling algorithms for some nonconvex families. *CoRR*, abs/0904.0583, 2009. arXiv:0904.0583.

10 B. Cousins and S. Vempala. A cubic algorithm for computing Gaussian volume. In *Proc. Symp. on Discrete Algorithms*, pages 1215–1228. SIAM/ACM, 2014.

11 K. Daniel and T.J. Moskowitz. Momentum crashes. *J. Financial Economics*, 122(2):221–247, 2016.

12 H.A. David. *Order statistics*. Wiley, New York, 2 edition, 1981.

13 L. Devroye. *Non-uniform Random Variate Generation*. Springer, Berlin, 1986.

14 A. Domahidi, E. Chu, and S. Boyd. ECOS: An SOCP solver for embedded systems. In *Proc. European Control Conference (ECC)*, pages 3071–3076, 2013.

15 M. Lo Duca, A. Koban, M. Basten, E. Bengtsson, B. Klaus, P. Kusmierczyk, J.H. Lang, C. Detken, and T. Peltonen. A new database for financial crises in european countries. Technical Report 13, European Central Bank and European Systemic Risk Board, Frankfurt am Main, Germany, 2017.

16 M. Dyer, A. Frieze, and R. Kannan. A random polynomial-time algorithm for approximating the volume of convex bodies. *J. ACM*, 38(1):1–17, 1991. doi:10.1145/102782.102783.

17 M.E. Dyer and A.M. Frieze. On the complexity of computing the volume of a polyhedron. *SIAM J. Comput.*, 17(5):967–974, 1988. doi:10.1137/0217060.

18 G. Elekes. A geometric inequality and the complexity of computing volume. *Discrete & Computational Geometry*, 1:289–292, 1986.

19 I.Z. Emiris and V. Fisikopoulos. Practical polytope volume approximation. *ACM Trans. Math. Soft.*, 2018. To appear. Prelim. version: Proc. Sympos. on Comput. Geometry, 2014.

20 E. Erdougan and G. Iyengar. An active set method for single-cone second-order cone programs. *SIAM J. Optimization*, 17(2):459–484, 2006. doi:10.1137/040612592.

21 J. Griffin, X. Ji, and J. Martin. Momentum investing and business cycle risk: Evidence from pole to pole. *J. Finance*, 58(6):2515–2547, 2003.

22 C. Grimme. Picking a uniformly random point from an arbitrary simplex. Technical report, Information Systems and Statistics, Munster U., Germany, 2015.

23 N. Jegadeesh and S. Titman. Returns to buying winners and selling losers: Implications for stock market efficiency. *J. Finance*, 48:65–91, 1993.

24 J. Lawrence. Polytope volume computation. *Math. of Computation*, 57(195):259–271, 1991.

25 O. Ledoit and M. Wolf. Honey, I shrunk the sample covariance matrix. *J. Portfolio Management*, 30(4):110–119, 2004. doi:10.3905/jpm.2004.110.

26 Y.T. Lee and S.S. Vempala. Convergence rate of Riemannian Hamiltonian Monte Carlo and faster polytope volume computation. *CoRR*, abs/1710.06261, 2017. arXiv:1710.06261.

27 Y.T. Lee and S.S. Vempala. Geodesic walks in polytopes. In *Proc. ACM Symp. on Theory of Computing*, pages 927–940. ACM, 2017. doi:10.1145/3055399.3055416.

28 L. Lovász. Hit-and-run mixes fast. *Math. Programming*, 86:443–461, 1999. doi:10.1007/s101070050099.

29 H. Markowitz. Portfolio selection. *J. Finance*, 7(1):77–91, 1952. doi:10.1111/j.1540-6261.1952.tb01525.x.

30 A.M. Mathai. On linear combinations of independent exponential variables. *Communications in Statistics: Theory & Methods*, 2007.

31 A.A. Mohsenipour and S.B. Provost. On approximating the distribution of quadratic forms in gamma random variables and exponential order statistics. *J. Statistical Theory & Appl.*, 12(2):173–184, 2013.

32 I. Pouchkarev. *Performance evaluation of constrained portfolios*. PhD thesis, Erasmus Research Institute of Management, The Netherlands, 2005.

33 I. Pouchkarev, J. Spronk, and J Trinidad. Dynamics of the spanish stock market through a broadband view of the IBEX 35 index. *Estudios Econom. Aplicada*, 22(1):7–21, 2004.

34 R.Y. Rubinstein and D.P. Kroese. *Simulation and the Monte Carlo method*. Wiley Interscience, New York, 2007.

35 R.Y. Rubinstein and B. Melamed. *Modern simulation and modeling*. Wiley, New York, 1998.

36 P. Scheurle and K. Spremann. Size, book-to-market, and momentum during the business cycle. *Review of Managerial Science*, 4(3):201–215, 2010.

37 N.A. Smith and R.W. Tromble. Sampling uniformly from the unit simplex. Technical report, Center for Language and Speech Processing, Johns Hopkins U., 2004.

38 G. Varsi. The multidimensional content of the frustum of the simplex. *Pacific J. Math.*, 46:303–314, 1973.

Subquadratic Encodings for Point Configurations

Jean Cardinal[1]
Département d'Informatique, Université libre de Bruxelles (ULB)
Brussels, Belgium
jcardin@ulb.ac.be
https://orcid.org/0000-0002-2312-0967

Timothy M. Chan
Department of Computer Science, University of Illinois at Urbana-Champaign
Champaign, IL, USA
tmc@illinois.edu

John Iacono[2]
Département d'Informatique, Université libre de Bruxelles (ULB)
Brussels, Belgium
socg18@johniacono.com
https://orcid.org/0000-0001-8885-8172

Stefan Langerman[3]
Département d'Informatique, Université libre de Bruxelles (ULB)
Brussels, Belgium
slanger@ulb.ac.be
https://orcid.org/0000-0001-6999-3088

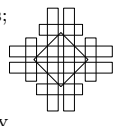
Aurélien Ooms[4]
Département d'Informatique, Université libre de Bruxelles (ULB)
Brussels, Belgium
aureooms@ulb.ac.be
https://orcid.org/0000-0002-5733-1383

Abstract

For many algorithms dealing with sets of points in the plane, the only relevant information carried by the input is the combinatorial configuration of the points: the orientation of each triple of points in the set (clockwise, counterclockwise, or collinear). This information is called the *order type* of the point set. In the dual, realizable order types and abstract order types are combinatorial analogues of line arrangements and pseudoline arrangements. Too often in the literature we analyze algorithms in the real-RAM model for simplicity, putting aside the fact that computers as we know them cannot handle arbitrary real numbers without some sort of encoding. Encoding an order type by the integer coordinates of a realizing point set is known to yield doubly exponential coordinates in some cases. Other known encodings can achieve quadratic space or fast orientation queries, but not both. In this contribution, we give a compact encoding for abstract order types that allows efficient query of the orientation of any triple: the encoding uses $O(n^2)$ bits and an orientation query takes $O(\log n)$ time in the word-RAM model with word size $w \geq \log n$. This encoding is space-optimal for abstract order types. We show how to shorten the encoding to $O(n^2(\log\log n)^2/\log n)$ bits for realizable order types, giving the first subquadratic

[1] Supported by the "Action de Recherche Concertée" (ARC) COPHYMA, convention number 4.110.H.000023.
[2] Supported by NSF grants CCF-1319648, CCF-1533564, CCF-0430849 and MRI-1229185, a Fulbright Fellowship and by the Fonds de la Recherche Scientifique-FNRS under Grant n° MISU F 6001 1 and two Missions Scientifiques.
[3] Directeur de recherches du Fonds de la Recherche Scientifique-FNRS.
[4] Supported by the Fund for Research Training in Industry and Agriculture (FRIA).

34th International Symposium on Computational Geometry (SoCG 2018).
Editors: Bettina Speckmann and Csaba D. Tóth; Article No. 20; pp. 20:1–20:14



Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

encoding for those order types with fast orientation queries. We further refine our encoding to attain $O(\log n / \log \log n)$ query time at the expense of a negligibly larger space requirement. In the realizable case, we show that all those encodings can be computed efficiently. Finally, we generalize our results to the encoding of point configurations in higher dimension.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases point configuration, order type, chirotope, succinct data structure

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.20

Related Version A full version of this paper is available at `https://arxiv.org/abs/1801.01767`

1 Introduction

At SoCG'86, Chazelle asked [29]:

> "How many bits does it take to know an order type?"

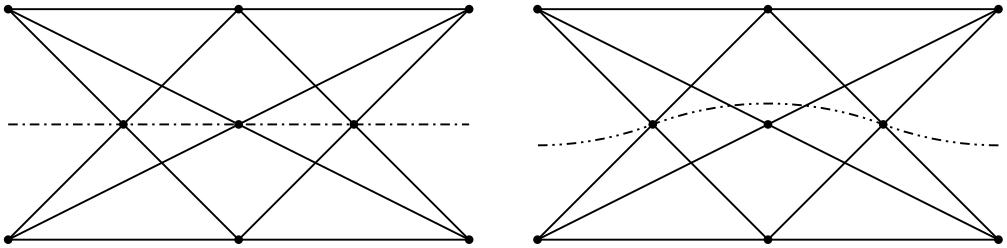
This question is of importance in Computational Geometry for the following two reasons: First, in many algorithms dealing with sets of points in the plane, the only relevant information carried by the input is the combinatorial configuration of the points given by the orientation of each triple of points in the set (clockwise, counterclockwise, or collinear) [18]. Second, computers as we know them can only handle numbers with finite description and we cannot assume that they can handle arbitrary real numbers without some sort of encoding. The study of *robust* algorithms is focused on ensuring the correct solution of problems on finite precision machines. Chapter 41 of The Handbook of Discrete and Computational Geometry is dedicated to this issue [41].

The (counterclockwise) orientation $\nabla(p, q, r) \in \{-, 0, +\}$ of a triple of points p, q, and r with coordinates (x_p, y_p), (x_q, y_q), and (x_r, y_r) is the sign of the determinant

$$\begin{vmatrix} 1 & x_p & y_p \\ 1 & x_q & y_q \\ 1 & x_r & y_r \end{vmatrix}.$$

Given a set of n labeled points $P = \{p_1, p_2, \ldots, p_n\}$, we define the *order type* of P to be the function $\chi\colon [n]^3 \to \{-, 0, +\}\colon (a, b, c) \mapsto \nabla(p_a, p_b, p_c)$ that maps each triple of point labels to the orientation of the corresponding points, up to isomorphism. A great deal of the literature in computational geometry deals with this notion [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 19, 20, 21, 22, 25, 26, 27, 28, 29, 30, 32, 33, 34, 36, 37, 38, 39]. The order type of a point set has been further abstracted into combinatorial objects known as (rank-three) *oriented matroids* [21]. The *chirotope axioms* define consistent systems of signs of triples [12]. From the topological representation theorem [13], all such *abstract* order types correspond to pseudoline arrangements, while, from the standard projective duality, order types of point sets correspond to straight line arrangements. See Chapter 6 of The Handbook for more details [38].

When the order type of a pseudoline arrangement can be realized by an arrangement of straight lines, we call the pseudoline arrangement *stretchable*. As an example of a nonstretchable arrangement, Levi gives Pappus's configuration where eight triples of concurrent straight lines force a ninth, whereas the ninth triple cannot be enforced by pseudolines [33] (see Figure 1). Ringel shows how to convert the so-called "non-Pappus" arrangement of Figure 1 (b)



(a) Realizable order type. **(b)** Abstract order type which is not realizable.

Figure 1 Pappus's configuration.

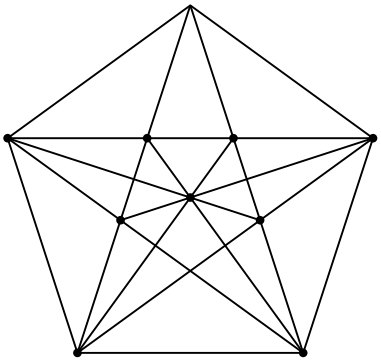


Figure 2 Perles's configuration.

to a simple arrangement while preserving nonstretchability [39]. All arrangements of eight or fewer pseudolines are stretchable [24], and the only nonstretchable simple arrangement of nine pseudolines is the one given by Ringel [37]. More information on pseudoline arrangements is available in Chapter 5 of The Handbook [23].

Figure 1 shows that not all pseudoline arrangements are stretchable. Indeed, most are not: there are $2^{\Theta(n^2)}$ abstract order types [19] and only $2^{\Theta(n \log n)}$ realizable order types [9, 27]. This discrepancy stems from the algebraic nature of realizable order types, as illustrated by the main tool used in the upper bound proofs (the Milnor-Thom Theorem [35, 40]).

Information theory implies that we need quadratic space for abstract order types whereas we only need linearithmic space for realizable order types. Hence, storing all $\binom{n}{3}$ orientations in a lookup table seems wasteful. Another obvious idea for storing the order type of a point set is to store the coordinates of the points, and answer orientation queries by computing the corresponding determinant. While this should work in many practical settings, it cannot work for all point sets. Perles's configuration shows that some configuration of points, containing collinear triples, forces at least one coordinate to be irrational [31](see Figure 2). Order types of points in general position can always be represented by rational coordinates. It is well known, however, that some configurations require doubly exponential coordinates, hence coordinates with exponential bitsizes if represented in the normal way [30].

Goodman and Pollack defined λ-matrices which can encode abstract order types using $O(n^2 \log n)$ bits [25]. They asked if the space requirements could be moved closer to the information-theoretic lower bounds. Felsner and Valtr showed how to encode abstract order types optimally in $O(n^2)$ bits via the wiring diagram of their corresponding allowable sequence [19, 20] (as defined in [22]). Aloupis et al. gave an encoding of size $O(n^2)$ that can be computed in $O(n^2)$ time and that can be used to test for the isomorphism of two

distinct point sets in the same amount of time [10]. However, it is not known how to decode the orientation of one triple from any of those encodings in, say, sublinear time. Moreover, since the information-theoretic lower bound for realizable order types is only $\Omega(n \log n)$, we must ask if this space bound is approachable for those order types while keeping orientation queries reasonably efficient.

Our results

In this contribution, we are interested in *compact* encodings for order types: we wish to design data structures using as few bits as possible that can be used to quickly answer orientation queries of a given abstract or realizable order type. In Section 2, we give the first optimal encoding for abstract order types that allows efficient query of the orientation of any triple: the encoding is a data structure that uses $O(n^2)$ bits of space with queries taking $O(\log n)$ time in the word-RAM model with word size $w \geq \log n$. Our encoding is far from being space-optimal for realizable order types. We show that its construction can be easily tuned to only require $O(n^2(\log\log n)^2/\log n)$ bits in this case. In Section 3, we further refine our encoding to reduce the query time to $O(\log n/\log\log n)$. In the realizable case, we give quadratic upper bounds on the preprocessing time required to compute an encoding in the real-RAM model. In the full version of the paper, we generalize our encodings for chirotopes of point sets in higher dimension [15].

Our data structure is the first subquadratic encoding for realizable order types that allows efficient query of the orientation of any triple. It is not known whether a subquadratic constant-degree algebraic decision tree exists for the related problem of deciding whether a point set contains a collinear triple. Any such decision tree would yield another subquadratic encoding for realizable order types. We see the design of compact encodings for realizable order types as a subgoal towards subquadratic nonuniform algorithms for this related problem, a major open problem in Computational Geometry. Note that pushing the preprocessing time below quadratic would yield such an algorithm.

2 Encoding order types via hierarchical cuttings

To make our statements clear, we use the following definition:

▶ **Definition 1.** For fixed k and given a function $f : [n]^k \to [O(1)]$, we define a $(S(n), Q(n))$-encoding of f to be a string of $S(n)$ bits such that, given this string and any $t \in [n]^k$, we can compute $f(t)$ in $Q(n)$ query time in the word-RAM model with word size $w \geq \log n$.

In this section, we use this definition with f being some order type,[5] $k = 3$ and the codomain of f being $\{-, 0, +\}$. For the rest of the discussion, we assume the word-RAM model with word size $w \geq \log n$ and the standard arithmetic and bitwise operators. We prove our main theorems for the two-dimensional case:

▶ **Theorem 2.** *All abstract order types have an $(O(n^2), O(\log n))$-encoding.*

▶ **Theorem 3.** *All realizable order types have an $(O(\frac{n^2(\log\log n)^2}{\log n}), O(\log n))$-encoding.*

[5] Technically, we encode the orientation predicate of some realizing arrangement of the order type and skip the isomorphism. If desired, a canonical labeling of the arrangement can be produced in $O(n^2)$ time for abstract and realizable order types [10].

▶ **Theorem 4.** *In the real-RAM model and the constant-degree algebraic decision tree model, given n real-coordinate input points in $\mathbb{R}^2$ we can compute the encoding of their order type as in Theorems 2 and 3 in $O(n^2)$ time.*

For instance, Theorem 3 implies that for any set of points $\{p_1, p_2, \ldots, p_n\}$, there exists a string of $O(n^2(\log\log n)^2/\log n)$ bits such that given this string and any triple of indices $(a, b, c) \in [n]^3$ we can compute the value of $\chi(a, b, c) = \nabla(p_a, p_b, p_c)$ in $O(\log n)$ time.

Throughout the rest of this paper, we assume that we can access some arrangement of lines or pseudolines that realizes the order type we want to encode. We thus exclusively focus on the problem of encoding the order type of a given arrangement. This does not pose a threat against the existence of an encoding. However, we have to be more careful when we bound the preprocessing time required to compute such an encoding. This is why, in Theorem 4, we specify the model of computation and how the input is given.

Hierarchical cuttings

We encode the order type of an arrangement via hierarchical cuttings as defined in [16]. A cutting in $\mathbb{R}^d$ is a set of (possibly unbounded and/or non-full dimensional) constant-complexity cells that together partition $\mathbb{R}^d$. A $\frac{1}{r}$-cutting of a set of n hyperplanes is a cutting with the constraint that each of its cells is intersected by at most $\frac{n}{r}$ hyperplanes. There exist various ways of constructing $\frac{1}{r}$-cuttings of size $O(r^d)$. Those cuttings allow for efficient divide-and-conquer solutions to many geometric problems. The hierarchical cuttings of Chazelle have the additional property that they can be composed without multiplying the hidden constant factors in the big-oh notation. In particular, they allow for $O(n^d)$-space $O(\log n)$-query d-dimensional point location data structures (for constant d). In the plane, hierarchical cuttings can be constructed for arrangement of pseudolines with the same properties.

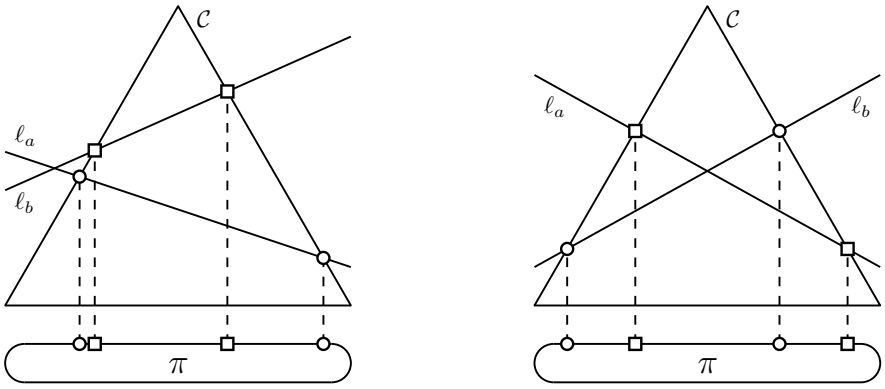
Idea

We want to preprocess n pseudolines $\{\ell_1, \ell_2, \ldots, \ell_n\}$ in the plane so that, given three indices a, b, and c, we can compute their orientation, that is, whether the intersection $\ell_a \cap \ell_b$ lies above, below or on ℓ_c. Our data structure builds on cuttings as follows: Given a cutting Ξ and the three indices, we can locate the intersection of ℓ_a and ℓ_b with respect to Ξ. The location of this intersection is a cell of Ξ. The next step is to decide whether ℓ_c lies above, lies below, contains or intersects that cell. In the first three cases, we are done. Otherwise, we can answer the query by recursing on the subset of pseudolines intersecting the cell containing the intersection. We build on hierarchical cuttings to solve all subproblems efficiently.

Intersection location

When the ℓ_a are straight lines, locating the intersection $\ell_a \cap \ell_b$ in Ξ is trivial if we know the real parameters of ℓ_a and ℓ_b and of the descriptions of the subcells of Ξ. However, in our model we are not allowed to store real numbers. To circumvent this annoyance, and to handle arrangements of pseudolines, we make a simple observation illustrated by Figure 3.

▶ **Observation 5.** *Two pseudolines ℓ_a and ℓ_b intersect in the interior of a full-dimensional cell $\mathcal{C}$ if and only if each pseudoline properly intersects the boundary of $\mathcal{C}$ exactly twice and their intersections with its boundary alternate.*



(a) $\ell_a \cap \ell_b \cap \mathcal{C} = \emptyset$. **(b)** $\ell_a \cap \ell_b \cap \mathcal{C} \neq \emptyset$.

Figure 3 Cyclic permutations (π).

This gives us a way to encode the location of the intersection of ℓ_a and ℓ_b in Ξ using only bits. For an arrangement of pseudolines, we use the standard vertical decomposition to construct a hierarchical cutting, which guarantees that a pseudoline intersects a cell boundary at most twice. For an arrangement of lines, we can use the standard bottom-vertex triangulation instead, which allows us to generalize our results to higher dimensions [15]. In the plane, the bottom-vertex triangulation partitions the space into triangular cells. We define the *cyclic permutation* of a full-dimensional cell $\mathcal{C}$ and a finite set of pseudolines $\mathcal{L}$ to be the finite sequence of properly intersecting pseudolines from $\mathcal{L}$ encountered when walking along the boundary of $\mathcal{C}$ in clockwise or counterclockwise order, up to rotation and reversal.

Note that non-full-dimensional cells are easier to encode. For a 0-dimensional cell and a pseudoline, we store whether the pseudoline lies above, lies below, or contains the 0-dimensional cell. For a 1-dimensional cell, a pseudoline could also intersect the interior of the cell, but in only one point. The intersections with that cell define an (acyclic) permutation with potentially several intersections at the same position. This information suffices to answer location queries for those cells, and the space taken is not more than that necessary for full-dimensional cells. When two pseudolines intersect in a 1-dimensional cell or contain the same 0-dimensional cell, they appear simultaneously in the cyclic permutation of an adjacent 2-dimensional cell if they intersect its interior. If that is the case, the location of the intersection of those two pseudolines in the cutting is the non-full-dimensional cell. A constant number of bits can be added to the encoding each time we need to know the dimension of the cell we encode.

Encoding

Given n pseudolines in the plane and some fixed parameter r, compute a hierarchical $\frac{1}{r}$-cutting of those pseudolines. This hierarchical cutting consists of ℓ levels labeled $0, 1, \ldots, \ell-1$. Level i has $O(r^{2i})$ cells. Each of those cells is further partitioned into $O(r^2)$ subcells. The $O(r^{2(i+1)})$ subcells of level i are the cells of level $i+1$. Each cell of level i is intersected by at most $\frac{n}{r^i}$ pseudolines, and hence each subcell is intersected by at most $\frac{n}{r^{i+1}}$ pseudolines.

We compute and store a combinatorial representation of the hierarchical cutting as follows: For each level of the hierarchy, for each cell in that level, for each pseudoline intersecting that cell, for each subcell of that cell, we store two bits to indicate the location of the pseudoline with respect to that subcell, that is, whether the pseudoline lies above (`00`), lies below (`01`)

or intersects the interior of that subcell (10). When a subcell is non-full-dimensional, we use another value (11) when the pseudoline contains the subcell. When a pseudoline intersects the interior of a 2-dimensional subcell, we also store the two indices of the intersections of that pseudoline with the subcell in the cyclic permutation associated with that subcell, beginning at an arbitrary location in, say, clockwise order. If the intersected subcell is 1-dimensional instead, we store the index of the intersection in the acyclic permutation associated with that subcell, beginning at an arbitrary endpoint. If two pseudolines intersect in the interior of a 1-dimensional subcell or on the boundary of a 2-dimensional subcell, they share the same index in the associated permutation.

This representation takes $O(\frac{n}{r^i} + \frac{n}{r^{i+1}} \log \frac{n}{r^{i+1}})$ bits per subcell of level i by storing for each pseudoline its location and, when needed, the permutation indices of its intersections with the subcell. For each of the $\lambda = O(\frac{n^2}{t^2})$ subcells of the last level of the hierarchy we store a pointer to a lookup table of size $\tau = O(t^3)$ that allows to answer the query of the orientation of any triple of pseudolines intersecting that subcell. The number $t = \frac{n}{r^\ell}$ denotes an upper bound on the number of pseudolines intersecting each of those subcells.

Storing the permutation at each subcell would suffice to answer all queries that do not reach the last level of the hierarchy. However, to get fast queries, we need to have access to all bits belonging to a given pseudoline without having to read the bits of the others. Using the Zone Theorem ([11, 17, 23]), and the fact that hierarchical cuttings are constructed by decompositions of subsets of the input pseudolines, we can bound the number of bits stored for a single pseudoline intersecting a given cell of level i by $\zeta_i = O(r^2 + r \log \frac{n}{r^{i+1}})$. This allows us to store all bits belonging to a given cell-pseudoline pair $(\mathcal{C}, \ell)$ in a contiguous block of memory $\sigma(\mathcal{C}, \ell)$ whose location in the encoding is easy to compute. We call $\sigma(\mathcal{C}, \ell)$ the *signature* of ℓ in $\mathcal{C}$. The overall number of bits stored stays the same up to a constant factor.

For queries that reach the last level of the hierarchy, storing an individual lookup table for each leaf would cost too much as soon as $t = \omega(1)$. However, as long as t is small enough, each order type is shared by many leaves, and we can thus reuse space. Formally, let $\nu(n)$ denote the number of order types of size n, which is $\nu(n) = 2^{\Theta(n^2)}$ for abstract order types and $\nu(n) = 2^{\Theta(n \log n)}$ for realizable order types. At most $\nu(t)$ distinct lookup tables are needed for answering the queries on the subcells of the last level of the hierarchy. Hence the pointers have size $\Psi = O(\log \nu(t))$ and the total size needed for the lookup tables is $O(t^3 \nu(t))$. For each leaf, we store a canonical labeling of size $\kappa = O(t \log t)$ on the pseudolines that intersect it. We use that canonical labeling to order the queries in the associated lookup table.

The encoding is the concatenation of the parameters n, r, and t, all signatures $\sigma(\mathcal{C}, \ell)$ for all pairs $(\mathcal{C}, \ell)$ with $\ell \cap C$, the canonical labelings at the leaves, the leaf pointers and the lookup tables. We define a canonical order on the cells of level i so that they can be ordered as $\{\mathcal{C}_{i,1}, \mathcal{C}_{i,2}, \ldots, \mathcal{C}_{i,O(r^{2i})}\}$. We complement the encoding with appropriate padding so that the position $\rho(\mathcal{C}_{i,j}, \ell)$ of the signature $\sigma(\mathcal{C}_{i,j}, \ell)$ is

$$\rho(\mathcal{C}_{i,j}, \ell) = \rho(\mathcal{C}_{i-1,1}, \ell_0^{\mathcal{C}_{i-1,1}}) + cr^{2i-2} \left\lfloor \frac{n}{r^{i-1}} \right\rfloor \zeta_{i-1} + (j-1) \left\lfloor \frac{n}{r^i} \right\rfloor \zeta_i + \pi(\mathcal{C}_i, \ell)\zeta_i,$$

where c is some constant, $\pi(\mathcal{C}, \ell)$ is the first index of ℓ in the cyclic permutation of $\mathcal{C}$, $\ell_a^{\mathcal{C}}$ is the pseudoline such that $\pi(\mathcal{C}, \ell_a^{\mathcal{C}}) = a$, and, for the root cell of the hierarchy $\mathcal{C}_{0,1}$ representing the entire space and containing all the intersections of the arrangement, $\rho(\mathcal{C}_{0,1}, \ell_0)$ is the position after the encoding of the parameters n, r, and t, and $\rho(\mathcal{C}_{0,1}, \ell_a) = \rho(\mathcal{C}_{0,1}, \ell_0) + a\zeta_0$.

The canonical labeling of the first leaf is stored at position

$$\rho_{\Lambda_0} = c\sum_{i=0}^{\ell-1} r^{2i} \left\lfloor \frac{n}{r^i} \right\rfloor \zeta_i,$$

the lookup table pointer of the first leaf is stored at position $\rho_{\Lambda_0} + \kappa$, the canonical labeling of leaf Λ is stored at position $\rho_{\Lambda_0} + (\Lambda - 1)(\kappa + \Psi)$ and its table lookup pointer at κ from that position. The first lookup table is stored at position $\rho_{\Lambda_0} + \lambda(\kappa + \Psi)$ and lookup table Θ is stored at position $\rho_{\Lambda_0} + \lambda(\kappa + \Psi) + (\Theta - 1)\tau$.

Space complexity

We prove a general bound on the space taken by our construction when the hierarchy contains ℓ levels. Let $H_r^\ell(n) \in \mathbb{N}$ be the maximum amount of space (bits), over all arrangements of n pseudolines, taken by the $\ell \in \mathbb{N}$ levels of a hierarchy with parameter $r \in (1, +\infty)$. This excludes the space taken by the lookup tables, their associated pointers and canonical labelings at the leaves, and the parameters of the hierarchy n, r and t.

▶ **Lemma 6.** *For $r \geq 2$ and $t = \frac{n}{r^\ell}$ we have*

$$H_r^\ell(n) = O\left(\frac{n^2}{t}(\log t + r)\right).$$

Proof. By definition, we have

$$H_r^\ell(n) = O\left(\sum_{i=0}^{\ell-1} \left(r^{2i} \cdot r^2 \cdot \left(\frac{n}{r^i} + \frac{n}{r^{i+1}} \log \frac{n}{r^{i+1}}\right)\right)\right).$$

We multiply the previous equation by $\frac{n}{tr^\ell} = 1$

$$H_r^\ell(n) = O\left(\frac{n^2}{t}\sum_{i=0}^{\ell-1} \left(\frac{1}{r^{\ell-i-1}} \cdot \left(r + \log \frac{n}{r^{i+1}}\right)\right)\right).$$

We use the equivalence $\frac{n}{r^{i+1}} = tr^{\ell-i-1}$ to replace the last term in the previous equation

$$H_r^\ell(n) = O\left(\frac{n^2}{t}\sum_{i=0}^{\ell-1} \left(\frac{1}{r^{\ell-i-1}} \cdot (r + \log t + (\ell - i - 1)\log r)\right)\right).$$

We reverse the summation by redefining $i \leftarrow \ell - i - 1$ and group the terms

$$H_r^\ell(n) = O\left(\frac{n^2}{t}\left((\log t + r)\sum_{i=0}^{\ell-1} \frac{1}{r^i} + \log r \sum_{i=0}^{\ell-1} \frac{i}{r^i}\right)\right).$$

Using the following inequalities:

$$\sum_{i=0}^{k} x^i \leq \frac{1}{1-x} \qquad \text{and} \qquad \sum_{i=0}^{k} ix^i \leq \frac{x}{(1-x)^2}, \qquad \forall k \in \mathbb{N}, \forall x \in (0,1),$$

we conclude that

$$H_r^\ell(n) = O\left(\frac{n^2}{t}\left(\left(1 + \frac{1}{r-1}\right)(\log t + r) + \left(1 + \frac{2r-1}{r^2 - 2r + 1}\right)\frac{\log r}{r}\right)\right),$$

and that for $r \geq 2$

$$H_r^\ell(n) = O\left(\frac{n^2}{t}(\log t + r)\right).$$

◀

Taking into account the space taken by the other bits of the encoding we obtain

▶ **Lemma 7.** *The space taken by our encoding is*

$$S_r^\ell(n) = O\left(\log ntr + \frac{n^2}{t}(\log t + r) + t^3\nu(t) + \frac{n^2}{t^2}(\log \nu(t) + t \log t)\right).$$

We pick r constant for both abstract and realizable order type. We have $\nu(t) = 2^{\Theta(n^2)}$ for abstract order types, hence we choose $t = \sqrt{\delta \log n}$ for small enough δ and the last term in Lemma 7 dominates with n^2. Note how the quadratic bottleneck of this encoding is the storage of the order type pointers at the leaves of the hierarchy. We have $\nu(t) = 2^{\Theta(n \log n)}$ for realizable order types, hence we choose $t = \delta \log n / \log\log n$ for small enough δ and the second and last term in Lemma 7 dominate with $n^2(\log\log n)^2/\log n$. This proves the space constraints in Theorems 2 and 3.

Correctness and query complexity

Given our encoding and three pseudoline indices a, b, c we answer a query as follows: We start by decoding the parameters n, r, and t. In our model, this can be done in $O(\log^* n + \log^* r + \log^* t)$ time.[6] Let $\mathcal{C} = \mathcal{C}_{0,1}$. First, find the subcell $\mathcal{C}'$ of $\mathcal{C}$ containing $\ell_a \cap \ell_b$ by testing for each subcell whether the intersections of ℓ_a and ℓ_b with the subcell alternate in the cyclic permutation. This can be done in $O(r^2)$ time by scanning $\sigma(\mathcal{C}, \ell_a)$ and $\sigma(\mathcal{C}, \ell_b)$ in parallel. Note that non-full dimensional subcells can be tested more easily. Next, if ℓ_c does not properly intersect $\mathcal{C}'$, answer the query accordingly. If on the other hand ℓ_c does properly intersect the subcell we recurse on $\mathcal{C}'$. This can be tested by scanning $\sigma(\mathcal{C}, \ell_c)$ in $O(r^2)$ time. Note that in case that the subcell is non-full-dimensional we can already answer the query. When we reach the relative interior of a subcell of the last level of the hierarchy without having found a satisfactory answer, we can answer the query by table lookup in constant time. This works as long as each order type identifier for at most t pseudolines fits in a constant number of words, which is the case for the values of t we defined. The position of the signatures scanned during the first recursive step of the query can be computed in constant time and at each other recursive step of the query we can compute the positions of the signatures we need to scan from the position of the signatures scanned during the previous recursive step in constant time. When we reach the bottom of the recursion, the position of the lookup table pointer, the position of the canonical labeling, and the position of the lookup table can be computed in constant time. The total query time is thus proportional to $r^2 \log_r n$ in the worst case, which is logarithmic since r is constant. This proves the query time constraints in Theorems 2 and 3. With the hope of getting faster queries we could pick $r = \Theta(\log t)$ to reduce the depth of the hierarchy, without changing the space requirements by more than a constant factor. However, if no additional care is taken, this would slow the queries down by a $\Theta(\log^2 t/\log\log t)$ factor because of the scanning approach taken when locating the intersection $\ell_a \cap \ell_b$. We show how to handle small but superconstant r properly in the next section.

[6] Logarithmic space and constant decoding time is trivial when $w = \Theta(\log n)$. If w is too large, encode n in binary using $\lceil \log n + 1 \rceil$ bits, $\lceil \log n + 1 \rceil$ using $\lceil \log \lceil \log n + 1 \rceil + 1 \rceil$ bits, $\lceil \log \lceil \log n + 1 \rceil + 1 \rceil$ using $\lceil \log \lceil \log \lceil \log n + 1 \rceil + 1 \rceil + 1 \rceil$ bits, etc. until the number to encode is smaller than a constant which we encode in unary with `1`'s. Prepend a `1` to the largest number and `0` to all the others except the smallest. Concatenate those numbers from smallest to largest. Total space is $O(\log n)$ bits and decoding n can be done in $O(\log^* n)$ time in the word-RAM model with $w \geq \log n$. As an alternative, logarithmic space and logarithmic decoding time is also trivially achievable with no constraint on w.

Preprocessing time

For a set of n points in the plane, or an arrangement of n lines in the dual, we can construct the encoding of their order type in quadratic time in the real-RAM and constant-degree algebraic computation tree models. We prove Theorem 4.

Proof. A hierarchical cutting can be computed in $O(nr^\ell)$ time in the dual plane. All signatures $\sigma(\mathcal{C}, \ell)$ can be computed from the cutting in the same time. The lookup tables and leaf-table pointers can be computed in $O(n^2 + t^3\nu(t))$ time as follows: For each subcell $\mathcal{C}$ among the $\frac{n^2}{t^2}$ subcells of the last level of the hierarchy, compute a canonical labeling and representation of the lines intersecting $\mathcal{C}$ in $O(t^2)$ time as in [10]. Insert the canonical representation in some trie in $O(t^2)$ time. If the canonical representation was not already in the trie, create a lookup table with the answers to all $O(t^3)$ queries on those lines and attach a pointer to that table in the trie. This happens at most $\nu(t)$ times. In the encoding, store the canonical labeling and this new pointer or the pointer that was already in the trie for the subcell $\mathcal{C}$. All parts of the encoding can be concatenated together in time proportional to the size of the encoding. ◀

3 Sublogarithmic query complexity

We further refine the data structure defined in the previous section so as to reduce the query time by a $\log\log n$ factor. We do so using specificities of the word-RAM model that allow us to preprocess computations on inputs of small but superconstant size. The idea is to make each signature $\sigma(\mathcal{C}, \ell)$ fit in a single word of memory by only approximately encoding the cyclic permutation of the intersections around each subcell, relying on the fact that ambiguous situations rarely arise. Those ambiguous situations, if they happen, can be deterministically handled using additional lookup tables. This improvement is applicable for both abstract and realizable order types.

We improve our main theorems for the two-dimensional case:

▶ **Theorem 8.** *All abstract order types have an* $(O(n^2), O(\frac{\log n}{\log\log n}))$*-encoding.*

▶ **Theorem 9.** *All realizable order types have a* $(O(\frac{n^2 \log^\varepsilon n}{\log n}), O(\frac{\log n}{\log\log n}))$*-encoding.*

▶ **Theorem 10.** *In the real-RAM model and the constant-degree algebraic decision tree model, given n real-coordinate input points in $\mathbb{R}^2$ we can compute the encoding of their order type as in Theorems 8 and 9 in $O(n^2)$ time.*

Bit packing

Fix a large α, a small δ and define $r = \Theta(\log^\delta n)$. Note that we can construct a hierarchical cutting with superconstant r by constructing a hierarchical cutting with some appropriate constant parameter r', and then skip levels that we do not need. Denote by $n_i = n/r^i$ an upper bound on the number of lines intersecting a cell of level i. For each subcell of level i, partition its cyclic permutation into $\log^\alpha n$ blocks of at most $n_{i+1}/\log^\alpha n$ intersections. For each pseudoline intersecting a cell we only store the block numbers that that pseudoline touches in its signature. Hence, each signature $\sigma(\mathcal{C}, \ell)$ only uses $\Theta(\log^{2\delta} n + \alpha \log^\delta n \log\log n) = \Theta(\log^{2\delta} n)$ bits, which fits in a word for small enough δ.

Intersection oracle

We construct an additional lookup table to compute the subcell in which $q_i \cap q_j$ lies in constant time. Computing it via scanning with so many subcells to check would waste any further savings. For that we need a general observation on the precomputation of functions on small universes.

▶ **Observation 11.** *In the word-RAM model with word size $w \geq \log n$, for any word-to-word function $f : [2^w] \to [2^w]$, we can build a lookup table of total bitsize $2^{s+1}w$ for all 2^s inputs $x \in [2^s]$ of bitsize $s \leq w$ in time $2^s T(s)$ where $T(s)$ is the complexity of computing $f(x)$, $x \in [2^s]$. The image of any input of bitsize s can then be retrieved in $O(1)$ time by a single lookup (since the input fits in a single word). In particular, we have $2^s T(s)$ and $2^{s+1}w$ sublinear as long as $T(s) = s^{O(1)}$ and $s \leq (1-\varepsilon)\log_2 n$.*

In other words, any polynomial time computable word-to-word function can be precomputed in sublinear time and space for all inputs of roughly logarithmic size.

Since our pseudoline identifiers now fit in $\Theta(\log^{2\delta} n)$ bits we can choose an appropriate δ so as to satisfy the requirements given above. We can thus precompute the function that sends two pseudoline identifiers to either the subcell containing their intersection or to some special value in case of an ambiguous input.

Disambiguation

Note that ambiguous inputs rarely occur. An input is ambiguous if and only if at least one boundary intersection of each pseudoline appears in the same cyclic permutation block of the cell that contains their intersection. This happens less than $\log^\alpha n \cdot (n_{i+1}/\log^\alpha n)^2 = \frac{n^2}{r^{2(i+1)}}/\log^\alpha n$ times per subcell of level i of the hierarchy. Summing over all levels, we get a subquadratic size lookup table for ambiguous cases which can be implemented using standard tools.

Space complexity

The total space used for the signatures $\sigma(\mathcal{C}, \ell)$ is proportional to

$$\sum_{i=0}^{\ell-1} r^{2i} \cdot r^2 \cdot \left(\frac{n}{r^i} + \frac{n}{r^{i+1}}\alpha \log\log n\right) = O\left(\frac{n^2}{t}(\log^\delta n + \alpha \log\log n)\right).$$

Intersection oracles and disambiguation tables fit in subquadratic space and the space analysis for the rest of the data structure still holds. For small enough δ the space remains quadratic for abstract order types and subquadratic for realizable order types. This proves the space constraints in Theorems 8 and 9. Unfortunately, we must incur a nonabsorbable extra $\log^\delta n$ factor in the realizable case. Note that a $\log\log\log n$ factor can be squeezed without increasing the space usage by choosing $r = \Theta(\log\log n)$ instead.

Correctness and query complexity

The previous analysis still holds modulo additional disambiguation lookups and oracle-based intersection location. We now have a shallower decision tree of depth $\log_r n = O_\delta(\frac{\log n}{\log\log n})$. This proves the query time constraints in Theorems 8 and 9.

Preprocessing time

We prove Theorem 10.

Proof. As before, the hierarchical cutting and all signatures $\sigma(C, \ell)$ can be computed in $O(nr^\ell)$ time. The lookup table and leaf-table pointers can be computed in $O(n^2)$ time. All intersection oracles and disambiguation tables can be computed in subquadratic time. ◀

4 Conclusion

Observe the following. Assume we are given an instance of some real-input decision problem. Given a decision tree of depth D for this problem for which each input query and answer can be encoded using at most Q bits, we can encode the instance using at most DQ bits by encoding the path traversed when executing the decision tree on this instance. The general position testing problem (GPT) asks if an input set of n points in the plane contains a collinear triple. It is an example of a real-input decision problem for which no subquadratic real-RAM algorithm is known even though the best known lower bound is only linearithmic. Since shallow decision trees yield short encodings, we see the design of a subquadratic encoding for realizable order types as a stepping stone towards nonuniform and uniform subquadratic algorithms for GPT.

Unfortunately, even though our encodings achieve subquadratic space for realizable order types, they cannot be used to test for isomorphism in subquadratic time. This is partly because the preprocessing time to construct the encoding is already quadratic. However, observe that the preprocessing time we achieve in this contribution matches the best known upper bound for GPT in the algebraic decision tree model:

▶ **Theorem 12.** *If there is an encoding with construction cost $C(N)$ for realizable order types in the algebraic decision tree model, then there is a nonuniform algorithm for general position testing that runs in time $C(N)$ in the algebraic decision tree model.*

Proof. Construct the encoding. Then at zero cost in the nonuniform model, run all $O(n^3)$ queries on the encoding. ◀

Goodman and Pollack saw GPT as a multidimensional generalization of sorting [25]. We again stress the need for a better understanding of this fundamental problem.

References

1. Oswin Aichholzer, Franz Aurenhammer, and Hannes Krasser. Enumerating order types for small point sets with applications. *Order*, 19(3):265–281, 2002.
2. Oswin Aichholzer, Franz Aurenhammer, and Hannes Krasser. On the crossing number of complete graphs. In *SOCG*, pages 19–24. ACM, 2002.
3. Oswin Aichholzer, Jean Cardinal, Vincent Kusters, Stefan Langerman, and Pavel Valtr. Reconstructing point set order types from radial orderings. *International Journal of Computational Geometry & Applications*, 26(3-4):167–184, 2016.
4. Oswin Aichholzer, Matias Korman, Alexander Pilz, and Birgit Vogtenhuber. Geodesic order types. *Algorithmica*, 70(1):112–128, 2014.
5. Oswin Aichholzer and Hannes Krasser. The point set order type data base: A collection of applications and results. In *CCCG*, pages 17–20, 2001.
6. Oswin Aichholzer and Hannes Krasser. Abstract order type extension and new results on the rectilinear crossing number. In *SOCG*, pages 91–98. ACM, 2005.

7 Oswin Aichholzer, Vincent Kusters, Wolfgang Mulzer, Alexander Pilz, and Manuel Wettstein. An optimal algorithm for reconstructing point set order types from radial orderings. In *ISAAC*, pages 505–516. Springer, 2015.

8 Oswin Aichholzer, Tillmann Miltzow, and Alexander Pilz. Extreme point and halving edge search in abstract order types. *Computational Geometry*, 46(8):970–978, 2013.

9 Noga Alon. The number of polytopes configurations and real matroids. *Mathematika*, 33(1):62–71, 1986.

10 Greg Aloupis, John Iacono, Stefan Langerman, Özgür Özkan, and Stefanie Wuhrer. The complexity of order type isomorphism. In *SODA*, pages 405–415. SIAM, 2014.

11 Marshall W. Bern, David Eppstein, Paul E. Plassmann, and F. Frances Yao. Horizon theorems for lines and polygons. In *Discrete and Computational Geometry*, volume 6 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 45–66. DIMACS/AMS, 1990.

12 Anders Björner, Michel Las Vergnas, Bernd Sturmfels, Neil White, and Günter M Ziegler. Oriented matroids. In *Encyclopedia of Mathematics*, volume 46. Cambridge University Press, 1993.

13 Jürgen Bokowski, Susanne Mock, and Ileana Streinu. On the Folkman-Lawrence topological representation theorem for oriented matroids of rank 3. *European Journal of Combinatorics*, 22(5):601–615, 2001.

14 Jürgen Bokowski, Jürgen Richter-Gebert, and Werner Schindler. On the distribution of order types. *Computational Geometry*, 1(3):127–142, 1992.

15 Jean Cardinal, Timothy M. Chan, John Iacono, Stefan Langerman, and Aurélien Ooms. Subquadratic encodings for point configurations. *ArXiv e-prints*, 2018. arXiv:1801.01767 [cs.CG].

16 Bernard Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete & Computational Geometry*, 9:145–158, 1993.

17 Bernard Chazelle, Leonidas J. Guibas, and D. T. Lee. The power of geometric duality. *BIT*, 25(1):76–90, 1985.

18 Herbert Edelsbrunner. *Algorithms in Combinatorial Geometry*, volume 10. Springer Science & Business Media, 2012.

19 Stefan Felsner. On the number of arrangements of pseudolines. In *SOCG*, pages 30–37. ACM, 1996.

20 Stefan Felsner and Pavel Valtr. Coding and counting arrangements of pseudolines. *Discrete & Computational Geometry*, 46(3):405–416, 2011.

21 Jon Folkman and Jim Lawrence. Oriented matroids. *Journal of Combinatorial Theory, Series B*, 25(2):199–236, 1978.

22 Jacob E. Goodman. Proof of a conjecture of Burr, Grünbaum, and Sloane. *Discrete Mathematics*, 32(1):27–35, 1980.

23 Jacob E. Goodman. Pseudoline arrangements. In *Handbook of Discrete and Computational Geometry, 2nd Ed.*, pages 97–128. Chapman and Hall/CRC, 2004.

24 Jacob E. Goodman and Richard Pollack. Proof of Grünbaum's conjecture on the stretchability of certain arrangements of pseudolines. *Journal of Combinatorial Theory, Series A*, 29(3):385–390, 1980.

25 Jacob E. Goodman and Richard Pollack. Multidimensional sorting. *SIAM Journal on Computing*, 12(3):484–507, 1983.

26 Jacob E. Goodman and Richard Pollack. Semispaces of configurations, cell complexes of arrangements. *Journal of Combinatorial Theory, Series A*, 37(3):257–293, 1984.

27 Jacob E. Goodman and Richard Pollack. Upper bounds for configurations and polytopes in $\mathbb{R}^d$. *Discrete & Computational Geometry*, 1:219–227, 1986.

28 Jacob E. Goodman and Richard Pollack. The complexity of point configurations. *Discrete Applied Mathematics*, 31(2):167–180, 1991.

29 Jacob E. Goodman and Richard Pollack. Allowable sequences and order types in discrete and computational geometry. In *New Trends in Discrete and Computational Geometry*, pages 103–134. Springer, 1993.

30 Jacob E. Goodman, Richard Pollack, and Bernd Sturmfels. Coordinate representation of order types requires exponential storage. In *STOC*, pages 405–410. ACM, 1989.

31 Branko Grünbaum. *Convex Polytopes*. Springer, 2005.

32 Alfredo Hubard, Luis Montejano, Emiliano Mora, and Andrew Suk. Order types of convex bodies. *Order*, 28(1):121–130, 2011.

33 Friedrich Levi. Die teilung der projektiven ebene durch gerade oder pseudogerade. *Ber. Math.-Phys. Kl. Sächs. Akad. Wiss*, 78:256–267, 1926.

34 Yoshitake Matsumoto, Sonoko Moriyama, Hiroshi Imai, and David Bremner. Matroid enumeration for incidence geometry. *Discrete & Computational Geometry*, 47(1):17–43, 2012.

35 John Milnor. On the Betti numbers of real varieties. *Proceedings of the American Mathematical Society*, 15(2):275–280, 1964.

36 Jaroslav Nešetřil and Pavel Valtr. A Ramsey property of order types. *Journal of Combinatorial Theory, Series A*, 81(1):88–107, 1998.

37 Jürgen Richter. Kombinatorische realisierbarkeitskriterien für orientierte matroide. *Mitt. Math. Sem. Univ. Giessen*, 194:1–112, 1989.

38 Jürgen Richter-Gebert and Günter M. Ziegler. Oriented matroids. In *Handbook of Discrete and Computational Geometry, 2nd Ed.*, pages 129–151. Chapman and Hall/CRC, 2004.

39 Gerhard Ringel. Teilungen der ebene durch geraden oder topologische geraden. *Mathematische Zeitschrift*, 64(1):79–102, 1956.

40 René Thom. Sur l'homologie des variétés algébriques. In *Differential and Combinatorial Topology (A Symposium in Honor of Marston Morse)*, pages 255–265, 1965.

41 Chee K. Yap. Robust geometric computation. In *Handbook of Discrete and Computational Geometry, 2nd Ed.*, pages 927–952. Chapman and Hall/CRC, 2004.

Algorithms for Low-Distortion Embeddings into Arbitrary 1-Dimensional Spaces

Timothy Carpenter
Dept. of Computer Science & Engineering, The Ohio State University
Columbus, USA
carpenter.454@osu.edu

Fedor V. Fomin
Department of Informatics, University of Bergen
Norway
fomin@ii.uib.no

Daniel Lokshtanov
Department of Informatics, University of Bergen
Norway
daniello@ii.uib.no

Saket Saurabh
Institute of Mathematical Sciences
Chennai, India
saket@imsc.res.in

Anastasios Sidiropoulos
Computer Science Dept., University of Illinois at Chicago
USA
sidiropo@uic.edu

Abstract

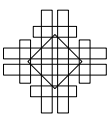
We study the problem of finding a minimum-distortion embedding of the shortest path metric of an unweighted graph into a "simpler" metric X. Computing such an embedding (exactly or approximately) is a non-trivial task even when X is the metric induced by a path, or, equivalently, the real line. In this paper we give approximation and fixed-parameter tractable (FPT) algorithms for minimum-distortion embeddings into the metric of a subdivision of some fixed graph H, or, equivalently, into any fixed 1-dimensional simplicial complex. More precisely, we study the following problem: For given graphs G, H and integer c, is it possible to embed G with distortion c into a graph homeomorphic to H? Then embedding into the line is the special case $H = K_2$, and embedding into the cycle is the case $H = K_3$, where K_k denotes the complete graph on k vertices. For this problem we give

- an approximation algorithm, which in time $f(H)\cdot\text{poly}(n)$, for some function f, either correctly decides that there is no embedding of G with distortion c into any graph homeomorphic to H, or finds an embedding with distortion $\text{poly}(c)$;
- an exact algorithm, which in time $f'(H, c) \cdot \text{poly}(n)$, for some function f', either correctly decides that there is no embedding of G with distortion c into any graph homeomorphic to H, or finds an embedding with distortion c.

Prior to our work, poly(OPT)-approximation or FPT algorithms were known only for embedding into paths and trees of bounded degrees.

2012 ACM Subject Classification Mathematics of computing → Approximation algorithms

Keywords and phrases Metric embeddings, minimum-distortion embeddings, 1-dimensional simplicial complex, Fixed-parameter tractable algorithms, Approximation algorithms



34th International Symposium on Computational Geometry (SoCG 2018).
Editors: Bettina Speckmann and Csaba D. Tóth; Article No. 21; pp. 21:1–21:14

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.21

Related Version A full version of the paper is available at `https://arxiv.org/abs/1712.06747`

1 Introduction

Embeddings of various metric spaces are a fundamental primitive in the design of algorithms [16, 18, 23, 22, 1, 2]. A low-distortion embedding into a low-dimensional space can be used as a sparse representation of a metrical data set (see e.g. [17]). Embeddings into 1- and 2-dimensional spaces also provide a natural abstraction of vizualization tasks (see e.g. [9]). Moreover embeddings into topologically restricted spaces can be used to discover interesting structures in a data set; for example, embedding into trees is a natural mathematical abstraction of phylogenetic reconstruction (see e.g. [11]). More generally, embedding into "algorithmically easy" spaces provides a general reduction for solving geometric optimization problems (see e.g. [7, 12]).

A natural algorithmic problem that has received a lot of attention in the past decade concerns the exact or approximate computation of embeddings of minimum distortion of a given metric space into some host space (or, more generally, into some space chosen from a specified family). Despite significant efforts, most known algorithms for this important class of problems work only for the case of the real line and trees.

In this work we present exact and approximate algorithms for computing minimum distortion embeddings into arbitrary 1-dimensional topological spaces of bounded complexity. More precisely, we obtain algorithms for embedding the shortest-path metric of a given unweighted graph into a subdivision of an arbitrary graph H. The case where H is just one edge is precisely the problem of embedding into the real line. We remark that prior to our work, even the case where H is a triangle, which corresponds to the problem of embedding into a cycle, was open.

We remark that the problem of embedding shortest path metrics of finite graphs into any fixed finite 1-dimensional simplicial complex $\mathcal{C}$ is equivalent to the problem of embedding into arbitrary subdivisions of some fixed finite graph H, where H is the abstract 1-dimensional simplicial complex corresponding to $\mathcal{C}$[1]. Since we are interested in algorithms, for the remainder of the paper we state all of our results as embeddings into subdivisions of graphs.

1.1 Our contribution

We now formally state our results and briefly highlight the key new techniques that we introduce. The input space consists of some unweighted graph G. The target space is some unknown subdivision H' of some fixed unweighted graph H; we allow the edges in H' to have arbitrary non-negative edge lengths.

We first consider the problem of approximating a minimum-distortion embedding into arbitrary H-subdivisions. We obtain a polynomial-time approximation algorithm, summarized in the following. The proof is given in Section 4.

[1] Here, a d-dimensional simplicial complex, for some integer $d \geq 1$, is the space obtained by taking a set of simplices of dimension at most d, and identifying pairs of faces of the same dimension. An abstract d-dimensional simplicial complex $\mathcal{A}$ is a family of nonempty subsets of cardinality at most $d+1$ of some ground set X, such that for all $Y' \subset Y \in \mathcal{A}$, we have $Y' \in \mathcal{A}$; in particular, any 1-dimensional simplicial complex corresponds to the set of edges and vertices of some graph.

▶ **Theorem 1.** *There exists a $8^h n^{\mathcal{O}(1)}$ time algorithm that takes as input an n-vertex graph G, a graph H on h vertices, and an integer c, and either correctly concludes that there is no c-embedding of G into any subdivision of H, or produces a c_{ALG}-embedding of G into a subdivision of H, with $c_{\mathsf{ALG}} \leq 64 \cdot 10^6 \cdot c^{24}(h+1)^9$.*

In addition, we also obtain a FPT algorithm, parameterized by the optimal distortion and H.

▶ **Theorem 2.** *There exists a $f(h,c) \cdot n^{\mathcal{O}(1)}$ time algorithm that takes as input an n-vertex graph H, a graph H on h vertices, and an integer c, and either correctly concludes that there is no c-embedding of G into any subdivision of H, or produces a c-embedding of G into a subdivision of H.*

1.2 Related work

Embedding into 1-dimensional spaces. Most of the previous work on approximation and FPT algorithms for low-distortion embedding (with one notable recent exception [27]) concerns embeddings of a more general metric space M into the real line and trees. However, even in the case of embedding into the line, all polynomial time approximation algorithms make assumptions on the metric M such as having bounded spread (which is the ratio between the maximum and the minimum point distances in M) [3, 26] or being the shortest-path metric of an unweighted graph [5]. This happens for a good reason: as it was shown by Bădoiu *et al.* [3], computing the minimum line distortion is hard to approximate up to a factor polynomial in n, even when M is a weighted tree metric with spread $n^{\mathcal{O}(1)}$.

Most relevant to our approximation algorithm is the work of Bădoiu *et al.* [5], who gave an algorithm that for a given n-vertex (unweighted) graph G and $c > 0$ in time $\mathcal{O}(cn^3)$ either concludes correctly that no c-distortion of G into line exists, or computes an $\mathcal{O}(c)$-embedding of G into the line. Similar results can be obtained for embedding into trees [5, 6]. Our approximation algorithm can be seen as an extension of these results to much more general metrics.

Parameterized complexity of low-distortion embeddings was considered by Fellows *et al.* [13], who gave an FPT algorithm for finding an embedding of an unweighted graph metric into the line with distortion at most c, or concludes that no such embedding exists, which works in time $\mathcal{O}(nc^4(2c+1)^{2c})$. As it was shown by Lokshtanov *et al.* [24], unless the exponential time hypothesis fails, this bound is asymptotically tight. For weighted graph metrics Fellows *et al.* obtained an algorithm with running time $\mathcal{O}(n(cW)^4(2c+1)^{2cW})$, where W is the largest edge weight of the input graph. In addition, they rule out, unless P=NP, any possibility of an algorithm with running time $\mathcal{O}((nW)^{h(c)})$, where h is a function of c alone. The problem of low-distortion embedding into a tree is FPT parameterized by the maximum vertex degree in the tree and the distortion c [13].

Due to the intractability of low-distortion embedding problems from the approximation and parameterized complexity perspective, Nayyeri and Raichel [26] initiated the study of approximation algorithms with running time, in the worst case, not necessarily polynomial and not even FPT. In a very recent work Nayyeri and Raichel [27] obtained a $(1+\varepsilon)$-approximation algorithm for finding the minimum-distortion embedding of an n-point metric space M into the shortest path metric space of a weighted graph H with m vertices. The running time of their algorithm is $(c_{\mathsf{OPT}}\Delta)^{\omega \cdot \lambda \cdot (1/\varepsilon)^{\lambda+2} \cdot \mathcal{O}((c_{\mathsf{OPT}})^{2\lambda})} \cdot n^{\mathcal{O}(\omega)} \cdot m^{\mathcal{O}(1)}$, where Δ is the spread of the points of M, ω is the treewidth of H and λ is the doubling dimension of H. Our approximation and FPT algorithms and the exact algorithm of Nayyeri and Raichel are incomparable. Their algorithm is for more general metrics but runs in polynomial time only when the optimal distortion c_{OPT} is constant, even when H is a cycle. In contrast,

our approximation algorithm runs in polynomial time for any value of c_{OPT}. Moreover, the algorithm of Nayyeri and Raichel is (approximation) FPT with parameter c_{OPT} only when the spread Δ of M (which in the case of the unweighted graph metric is the diameter of the graph) and the doubling dimension of the host space are both constants; when $c_{\mathsf{OPT}} = \mathcal{O}(1)$ (which is the interesting case for FPT algorithms), this implies that the doubling dimension of M must also be constant, and therefore M can contain only a constant number of points, this makes the problem trivially solvable in constant time. The running time of our parameterized algorithm does not depend on the spread of the metric of M.

Embedding into higher dimensional spaces. Embeddings into d-dimensional Euclidean space have also been investigated. The problem of approximating the minimum distortion in this setting appears to be significantly harder, and most known results are lower bounds [25, 10]. Specifically, it has been shown by Matoušek and Sidiropoulos [25] that it is NP-hard to approximate the minimum distortion for embedding into $\mathbb{R}^2$ to within some polynomial factor. Moreover, for any fixed $d \geq 3$, it is NP-hard to distinguish whether the optimum distortion is at most α or at least n^β, for some universal constants $\alpha, \beta > 0$. The only known positive results are a $\mathcal{O}(1)$-approximation algorithm for embedding subsets of the 2-sphere into $\mathbb{R}^2$ [5], and approximation algorithms for embedding ultrametrics into $\mathbb{R}^d$ [4, 9].

Bijective embeddings. We note that the approximability of minimum-distortion embeddings has also been studied for the case of bijections [28, 15, 19, 21, 8, 20, 10]. In this setting, most known algorithms work for subsets of the real line and for trees.

2 Notation and definitions

For a graph G, we denote by $V(G)$ the set of vertices of G and by $E(G)$ the set of edges of G. For some $U \subseteq V(G)$, we denote by $G[U]$ the subgraph of G induced by U. Let $\mathsf{deg}_{\mathsf{max}}(G)$ denote the maximum degree of G.

Let $M = (X, d)$, $M' = (X', d')$ be metric spaces. An injective map $f : X \to X'$ is called an *embedding*. The *expansion* of f is defined to be $\mathsf{expansion}(f) = \sup_{x' \neq y' \in X} \frac{d'(f(x'), f(y'))}{d(x', y')}$ and the *contraction* of f is defined to be $\mathsf{contraction}(f) = \sup_{x \neq y \in X} \frac{d(x,y)}{d'(f(x), f(y))}$. We say that f is *non-expanding* (resp. *non-contracting*) if $\mathsf{expansion}(f) \leq 1$ (resp. $\mathsf{contraction}(f) \leq 1$). The distortion of f is defined to be $\mathsf{distortion}(f) = \mathsf{expansion}(f) \cdot \mathsf{contraction}(f)$. We say that f is a *c-embedding* if $\mathsf{distortion}(f) \leq c$.

For a metric space $M = (X, d)$, for some $x \in X$, and $r \geq 0$, we write $\mathsf{ball}_M(x, r) = \{y \in X : d(x, y) \leq r\}$, and for some $Y \subseteq X$, we define $\mathsf{diam}_M(Y) = \sup_{x,y \in Y} d(x, y)$. We omit the subscript when it is clear from the context. We also write $\mathsf{diam}(M) = \mathsf{diam}_M(X)$. When M is finite, the *local density* of M is defined to be $\delta(M) = \max_{x \in X, r > 0} \frac{|\mathsf{ball}_M(x,r)|-1}{2r}$. For a graph G, we denote by d_G the shortest-path distance in G. We shall often use G to refer to the metric space $(V(G), d_G)$.

For graphs H and H', we say that H' is a *subdivision* of H if it is possible, after replacing every edge of H by some path, to obtain a graph isomorphic to H'.

3 Overview of our results and techniques

Here we present our main theorems and algorithms, with a short discussion. Formal proofs and detailed statements of the algorithms can be either found in the later section or in the appended full version of the paper.

Approximation algorithm for embedding into an H-subdivision for general H. Here, we briefly highlight the main ideas of the approximation algorithm for embedding into H-subdivisions, for arbitrary fixed graph H. A key concept is that of a *proper* embedding: this is an embedding where every edge of the target space is "necessary". In other words, for every edge e of H' there exists some vertices u, v in G, such that the shortest path between u and v in H' traverses e. Embeddings that are not proper are difficult to handle. We first guess the set of edges in H such that their corresponding paths in H' contain unnecessary edges; we "break" those edges of H into two new edges, each having a leaf as one of their endpoint. There is a bounded number of guesses (depending on H), and we are guaranteed that for at least one guess, there exists an optimal embedding that is proper. By appropriately scaling the length of the edges in H' we may assume that the embedding we are looking for has contraction exactly 1. The importance of using proper embeddings is that a proper embedding which is "locally" non-contracting is also (globally) non-contracting, while this is not necessarily true for non-proper embeddings.

A second difficulty is that we do not know the number of times that an edge in H is being subdivided. Guessing the exact number of times each edge is subdivided would require $n^{f(H)}$ time, which is too much. Instead we set a specific threshold ℓ, based on c. The threshold ℓ is approximately c^3, and essentially ℓ is a threshold for how many vertices a BFS in G needs to see before it is able to distinguish between a part of G that is embedded on an edge, and a part of G that is embedded in an area of H' close to a vertex of degree at least 3. In particular, parts of G that are embedded close to the middle of an edge *can* be embedded with low distortion onto the line, while parts that are embedded close to a vertex of degree 3 in H can not - because G "grows in at least 3 different directions" in such parts. Since a BFS can be used as an approximation algorithm for embedding into the line, it will detect whether the considered part of G is close to a degree ≥ 3 vertex of H or not.

Instead of guessing exactly how many times each edge of H is subdivided, we guess for every edge whether it is subdivided at least ℓ times or not. The edges of H that are subdivided at least ℓ times are called "long", while the edges that are subdivided less than ℓ times are called "short". We call the connected components of H induced by the short edges a *cluster*. Having defined clusters, we now observe that a cluster with only two long edges leaving it *can* be embedded into the line with (relatively) low distortion, contradicting what we said in the previous paragraph! Indeed, the parts of G mapped to a cluster with only two long edges leaving it are (from the perspective of a BFS), indistinguishable from the parts that are mapped in the middle of an edge! For this reason, we classify clusters into two types: the *boring* ones that have at most two (long) edges leaving them, and the *interesting* ones that are incident to at least 3 long edges.

Any graph can be partitioned into vertices of degree at least 3 and paths between these vertices such that every internal vertex on these paths has degree 2. Thinking of clusters as "large" vertices and the long edges as edges between clusters, we can now partition the "cluster graph" into interesting clusters (analogs of the vertices of degree at least 3), and chains of boring clusters between the interesting clusters (analogs of the paths of vertices of degree 2).

The parts of G that are embedded onto a chain of boring clusters can be embedded into the line with low distortion, and therefore, for a BFS these parts are indistinguishable from the parts of G that are embedded onto a single long edge. However, the interesting clusters are distinguishable from the boring ones, and from the parts of G that are mapped onto long edges, because around interesting clusters the graph really does "grow in at least 3 different directions" for a long enough time for a BFS to pick up on this.

Using the insights above, we can find a set F of at most $|V(H)|$ vertices in G, such that every vertex in F is mapped "close" to some interesting cluster, and such that every interesting cluster has some vertex in F mapped "close" to it. At this point, one can essentially just guess in time $\mathcal{O}(h^h)$ which vertex of F is mapped close to which clusters of H. Then one maps each of the vertices that are "close" to F (in G) to some arbitrarily chosen spot in H which is close enough to the image of the corresponding vertex of F. Local density arguments show that there are not too many vertices in G that are "close" to F, and therefore this arbitrary choice will not drive the distortion of the computed mapping up too much.

It remains to embed all of the vertices that are "far" from F in G. However, by the choice of F we know that all such vertices should be embedded onto long edges, or onto chains of boring clusters. Thus, each of the yet un-embedded parts of the graph can be embedded with low distortion into the line! All that remains is to compute such low distortion embeddings for each part using a BFS, and assign each part to an edge of H. Stitching all of these embeddings together yields the approximation algorithm.

There are multiple important details that we have completely ignored in the above exposition. The most important one is that a cluster can actually be quite large when compared to a long edge. After all, a boring cluster contains up to $E(H)$ short edges, and the longest short edge can be almost as long as the shortest long edge! This creates several technical complications in the algorithm that computes the set F. Resolving these technical complications ends up making it unnecessary to guess which vertex of F is mapped to which vertex of H, instead one can compute this directly, at the cost of increasing the approximation ratio.

FPT algorithm for embedding into an H-subdivision for general H. Our FPT algorithm for embedding a graph G into H-subdivisions (for arbitrary fixed H) draws inspiration from the algorithm for the line used in [14, 5], while also using an approach similar to the approximation algorithm for H-subdivisions. The result here is an exact algorithm with running time $f(H, c_{\mathsf{OPT}}) \cdot n^{\mathcal{O}(1)}$. A naive generalization of the algorithm for the line needs to maintain the partial solution over $f(H)$ intervals, which results in running time $n^{g(H)}$, which is too much. Supposing that there is a proper c-embedding of G into some H-subdivision, we attempt to find this embedding by guessing the short and long edges of H. Using this guess, we partition H into connected clusters of short and long edges (we call the clusters of short edges "interesting" clusters, and the clusters of long edges "path" clusters). We show that if a c-embedding exists, we can find a subset of $V(G)$, with size bounded by a function of $|H|$ and c, that contains all vertices embedded into the interesting clusters of H. From this, we make further guesses as to which specific vertices are embedded into which interesting clusters, then how they are embedded into the interesting clusters. We also make guesses as to what the embedding looks like for a short distance (for example, $\mathcal{O}(c^2)$) along the long edges which are connected to the important clusters.

Since the number of guesses at each step so far can be bounded in terms of c and H, we can iterate over all possible configurations. Once our guesses have found the correct choices for the interesting clusters and for a short distance along the paths leaving these clusters, we are able to partition the remaining vertices of G, and guess which path clusters these partitions are embedded into. Due to the "path-like" nature of the path clusters, when we pair the correct partition and path cluster, we are able to use an approach inspired by [14, 5] to find a c-embedding of the partition into the path cluster, which is compatible with the choices already made for the interesting clusters.

4 An approximation algorithm for embedding into arbitrary graphs

In this section we give a complete (technical) description of our approximation algorithm for embedding into arbitrary graphs; albeit without proof sometimes. In particular, we prove Theorem 1. We start by formally defining the notions of pushing and proper embeddings.

4.1 Proper, pushing, and non-contracting embeddings

Let G, H be connected graphs, with a fixed total order $<$ on $V(G)$ and $V(H)$. A non-contracting, c_{OPT}-embedding of G to H is a function $f_{\mathsf{OPT}} : V(G) \to (H_{\mathsf{OPT}}, w_{\mathsf{OPT}})$, where H_{OPT} is a subdivision of H, $w_{\mathsf{OPT}} : E(H_{\mathsf{OPT}}) \to \mathbb{R}^{>0}$, and for all $u, v \in V(G)$,

$$d_G(u,v) \leq d_{(H_{\mathsf{OPT}}, w_{\mathsf{OPT}})}(f_{\mathsf{OPT}}(u), f_{\mathsf{OPT}}(v)) \leq c_{\mathsf{OPT}} \cdot d_G(u,v),$$

where $d_{(H_{\mathsf{OPT}}, w_{\mathsf{OPT}})}$ is the shortest path distance in H_{OPT} with respect to w_{OPT}. Stated formally, for all $h_1, h_2 \in V(H_{\mathsf{OPT}})$, if $\mathcal{P}$ is the set of all paths from h_1 to h_2 in H_{OPT}, then

$$d_{(H_{\mathsf{OPT}}, w_{\mathsf{OPT}})}(h_1, h_2) = \min_{P \in \mathcal{P}} \left\{ \sum_{e \in P} w_{\mathsf{OPT}}(e) \right\}.$$

▶ **Definition 3.** For a graph G_1, a subdivision G_1' of G_1, and and edge $e \in E(G_1)$, let $\mathsf{SUB}_{G_1'}(e)$ be the subdivision of e in G_1'. For convenience, for each $e \in E(H)$, we shall use e_{OPT} to indicate the subdivision of e in H_{OPT}.

The following notion of consecutive vertices will be necessary to describe additional properties we will want our embeddings to have.

▶ **Definition 4.** Suppose there exists $u, v \in V(G)$ and $e \in E(H)$ such that $f_{\mathsf{OPT}}(u), f_{\mathsf{OPT}}(v) \in V(e_{\mathsf{OPT}})$ and $f_{\mathsf{OPT}}(u) < f_{\mathsf{OPT}}(v)$. If for all $w \in V(G) \setminus \{u, v\}$, $f_{\mathsf{OPT}}(w)$ is not in the path in e_{OPT} between $f_{\mathsf{OPT}}(u)$ and $f_{\mathsf{OPT}}(v)$, then we say that u and v are *consecutive w.r.t. e*, or we say that u and v are *consecutive*.

The first property we will want our embeddings to have is that they are "pushing". The intuition here is that we want our embedding to be such that we cannot modify it by contracting the distance further between two consecutive vertices.

▶ **Definition 5.** If for all $u, v \in V(G)$ and $e \in E(H)$ such that u and v are consecutive w.r.t. e we have that $d_{e_{\mathsf{OPT}}}(f_{\mathsf{OPT}}(u), f_{\mathsf{OPT}}(v)) = d_G(u,v)$, then we say that f_{OPT} is *pushing*.

The next property we want for our embeddings is that they are "proper", meaning that all edges of the target graph are, in a loose sense, covered by an edge of the source graph.

▶ **Definition 6.** For any $z \in V(H_{\mathsf{OPT}})$, if there exists $\{u, v\} \in E(G)$ such that

$$d_{(H_{\mathsf{OPT}}, w_{\mathsf{OPT}})}(f_{\mathsf{OPT}}(u), f_{\mathsf{OPT}}(v)) = d_{(H_{\mathsf{OPT}}, w_{\mathsf{OPT}})}(z, f_{\mathsf{OPT}}(u)) + d_{(H_{\mathsf{OPT}}, w_{\mathsf{OPT}})}(z, f_{\mathsf{OPT}}(v))$$

then we say that z is *proper w.r.t. f_{OPT}*. If for all $x \in V(H_{\mathsf{OPT}})$, x is proper w.r.t. f_{OPT}, then we say that f_{OPT} is *proper*.

Given some target graph to embed into, there may not necessarily be a proper embedding. However, for some "quasi-subgraph" (defined below) of our target, there will be a proper embedding, which can be used to find an embedding into the target graph.

▶ **Definition 7.** Let J and J' be connected graphs. We say J' is a *quasi-subgraph* of J if J can be made isomorphic to J' by applying any sequence of the following rules to J: (1.) Delete a vertex in $V(J)$. (2.) Delete an edge in $E(J)$. (3.) Delete an edge $\{u, v\} \in E(J)$, add vertices u', v' to $V(J)$, and add edges $\{u, u'\}, \{v, v'\}$ to $E(J)$.

By examining the quasi-subgraphs of our target graph, we can restrict our search to proper, pushing, non-contracting embeddings. This leads to the following result.

▶ **Lemma 8.** *There exists a proper, pushing, non-contracting c_{OPT}-embedding of G to some (H^q, w^q), where H^q is the subdivision of some quasi-subgraph of H, and $w^q : E(H^q) \to \mathbb{R}^{>0}$.*

4.2 Approximation algorithm

In this subsection we give our approximation algorithm. By Lemma 8 there is a proper, pushing c_{OPT}-embedding of G into a subdivision of a quasi-subgraph H^q of H with edge weight function w^q. Furthermore, by subdividing each edge of H sufficiently many times, for any $\epsilon > 0$ any c-embedding of G into (H^q, w^q) can be turned into an $(c + \epsilon)$-embedding of G into a subdivision of H.

The weighted quasi-subgraph (H^q, w^q) of H is a subdivision of a quasi-subgraph H_{sub} of H. Since H has less than $3^{|E(H)|+|V(H)|}$ quasi-subgraphs our algorithm can guess H_{sub}. Thus, for the purposes of our approximation algorithm, it is sufficient to find an embedding of G into a weighted subdivision (H_{ALG}, w_{ALG}) of H_{sub} under the assumption that a proper, pushing c_{OPT}-embedding of G into some weighted subdivision of H_{sub} exists. Furthermore, any proper and pushing embedding is non-contracting and has contraction exactly equal to 1. Such an embedding f is a c-embedding if and only if for every edge

$$uv \in E(G), d_{(H,w)}(f(u), f(v)) \leq c. \tag{1}$$

Thus, to prove that our output embedding is a c-embedding (for some c) we will prove that it is proper, pushing and that (1) is satisfied. Thus, the main technical result of this section is encapsulated in the following lemma.

▶ **Lemma 9.** *There is an algorithm that takes as input a graph G with n vertices, a graph H and an integer c, runs in time $2^h \cdot n^{\mathcal{O}(1)}$ and either correctly concludes that there is no c-embedding of G into a weighted subdivision of H, or produces a proper, pushing c_{ALG}-embedding of G into a weighted subdivision of a quasi-subgraph H' of H, where $c_{\mathsf{ALG}} = \mathcal{O}(c^{24}h^9)$.*

Definitions. To prove Lemma 9 we need a few definitions. Throughout the section we will assume that there exists a weighted subdivision $(H_{\mathsf{OPT}}, w_{\mathsf{OPT}})$ and a c-embedding $f_{\mathsf{OPT}} : V(G) \to V(H_{\mathsf{OPT}})$. This embedding is unknown to the algorithm and will be used for analysis purposes only. Every edge $e = uv$ in H corresponds to a path P_e in H_{OPT} from u to v. Based on the embedding $f_{\mathsf{OPT}} : V(G) \to V(H_{\mathsf{OPT}})$ we define the *embedding pattern* function $\hat{f}_{\mathsf{OPT}} : V(G) \to V(H) \cup E(H)$ as follows. For every vertex $v \in V(G)$ such that f_{OPT} maps v to a vertex of H_{OPT} that is also a vertex of H, $\hat{f}_{\mathsf{OPT}}$ maps v to the same vertex. In other words if $f_{\mathsf{OPT}}(v) = u$ for $u \in V(H)$, then $\hat{f}_{\mathsf{OPT}}(v) = u$. Otherwise f_{OPT} maps v to a vertex u on a path P_e corresponding to an edge $e \in E(H)$. In this case we set $\hat{f}_{\mathsf{OPT}}(v) = e$.

We will freely make use of the "inverses" of the functions f_{OPT} and $\hat{f}_{\mathsf{OPT}}$. For a vertex set $C \subseteq V(H_{\mathsf{OPT}})$ we define $f^{-1}_{\mathsf{OPT}}(C) = \{v \in V(G) : f_{\mathsf{OPT}}(v) \in C\}$. We will also naturally extend functions that act on elements of a universe to subsets of that universe. For example, for a set $F \subseteq E(H_{\mathsf{OPT}})$ we use $w_{\mathsf{OPT}}(F)$ to denote $\sum_{e \in F} w_{\mathsf{OPT}}(e)$. We further extend this convention to write $w_{\mathsf{OPT}}(P_e)$ instead of $w_{\mathsf{OPT}}(E(P_e))$ for a path (or a subgraph) of H_{OPT}. We extend the distance function to also work for distances between sets.

Throughout the section we will use the following parameters, for now ignore the parenthesized comments to the definitions of the parameters, these are useful for remembering the purpose of the parameter when reading the proofs: $h = |E(H)|$ (the number of edges in H), c (the distortion of f_{OPT}), $\ell = 20c^3$ (long edge threshold), $r = 5\ell h$ (half of covering radius), and $c_{ALG} = 64 \cdot 10^6 \cdot c^{24}(h+1)^9$ (distortion of output embedding).

Using the parameter ℓ we classify the edges of H into short and long edges. An edge $e \in E(H)$ is called *short* if $w_{\mathsf{OPT}}(P_e) \leq \ell$ and it is called *long* otherwise. The edge sets E_{short} and E_{long} denote the set of short and long edges in H respectively. A *cluster* in H is a connected component C of the graph $H_{short} = (V(H), E_{short})$. We abuse notation and denote by C both the connected component and its vertex set. The *long edge degree* of a cluster C in H is the number of long edges in H incident to vertices in C. Here a long edge whose both endpoints are in C is counted twice. A cluster C of long edge degree at most 2 is called *boring*, otherwise it is *interesting*. Most of the time when discussing clusters, we will be speaking of clusters in H. However we overload the meaning of the word cluster to mean something else for vertex sets of G. A *cluster in* G is a set C such that there exists a cluster C_H of H such that $C = \{v \in V(G) \ : \ \hat{f}_{\mathsf{OPT}}(v) \in V(C_H) \cup E(C_H)\}$. Thus there is a one to one correspondence between clusters in G and H.

A *cluster-chain* is a sequence $C_1, e_1, C_2, e_2, \ldots, e_{t-1}, C_t$ such that the following conditions are satisfied. First, the C_i's are distinct clusters in H, except that possibly $C_1 = C_t$. Second, C_1 and C_t are interesting, while $C_2, \ldots C_{t-1}$ are boring. Finally, for every $i < t$ the edge e_i is a long edge in H connecting a vertex of C_i to a vertex of C_{i+1}.

4.2.1 Using Breadth First Search to detect interesting clusters

In this subsection we prove a lemma that is the main engine behind Lemma 9. Once the main engine is set up, all we will need to finish the proof of Lemma 9 will be to complete the embedding by running the approximation algorithm for embedding into a line for each cluster-chain of H, and stitching these embeddings together.

Before stating the lemma we define what it means for a vertex set F in G to cover a cluster. We say that a vertex set $F \subseteq V(G)$ *r-covers* a cluster C in G if some vertex in F is at distance at most r (in G) from at least one vertex in C. A vertex set $F \subseteq V(G)$ covers a cluster C in H if F covers the cluster C_G corresponding to C in G.

▶ **Lemma 10** (Interesting Cluster Covering Lemma)**.** *There exists an algorithm that takes as input G, H and c, runs in time $2^h n^{\mathcal{O}(1)}$ and halts. If there exists a proper c-embedding $\hat{f}_{\mathsf{OPT}}$ from G to a weighted subdivision of H, the algorithm outputs a family $\mathcal{F} \subset 2^{V(G)}$ such that $|\mathcal{F}| \leq 2^h$, every set $F \in \mathcal{F}$ has size at most h, and there exists an $F \in \mathcal{F}$ that $2r$-covers all interesting clusters of H.*

We do not prove Lemma 10 here, however we describe the algorithm used in Lemma 10. The algorithm iteratively adds vertices to a set F. During the iteration the algorithm makes some non-deterministic steps, these steps result in the algorithm returning a family of sets $\mathcal{F}$ rather than a single set F. We now describe a crucial subroutine of the algorithm of Lemma 10 that we call the SEARCH algorithm. The algorithm takes as input G, c, a set $F \subseteq V(G)$ and a vertex v. The algorithm explores the graph, starting from v with the aim of finding a local structure in G that on one hand can not be embedded into the line with low distortion, while on the other hand is far away from F. It will either output fail, meaning that the algorithm failed to find a structure not embeddable into the line, or success together with a vertex $\hat{u}$, meaning that the algorithm succeeded to find a structure not embeddable into the line, and that $\hat{u}$ is close to this structure.

Description of the SEARCH algorithm. The algorithm takes as input G, c, a set $F \subseteq V(G)$ and a vertex v. It performs a breadth first search (BFS) from v in G. Let X_1, X_2, etc. be the BFS layers starting from v. In other words $X_i = \{x \in V(G) \ : \ d_G(v,x) = i\}$. The algorithm inspects the BFS layers $X_1, X_2, \ldots$ one by one in increasing order of i.

For $i < 2c^2$ the algorithm does nothing other than the BFS itself. For $i = 2c^2$ the algorithm proceeds as follows. It picks an arbitrary vertex $v_L \in X_i$ and picks another vertex $v_R \in X_i$ at distance at least $2c+1$ from v_L in G. Such a vertex v_R might not exist, in this case the algorithm proceeds without picking v_R. The algorithm partitions X_i into X_i^L and X_i^R in the following way. For every vertex $x \in X_i$, if $d_G(x, v_L) \leq 2c$ then x is put into X_i^L. If $d_G(x, v_R) \leq 2c$ then x is put into X_i^R. If some vertex $x \in X_i$ is put *both* into X_i^L and in X_i^R, or *neither* into X_i^L nor into X_i^R the algorithm returns success together with $\hat{u} = v$.

For $i > 2c^2$ the algorithm proceeds as follows. If any vertex in X_i is at distance at most r from any vertex in F (in the graph G), the algorithm outputs fail and halts. Otherwise, the algorithm partitions X_i into X_i^L and X_i^R. A vertex $x \in X_i$ is put into X_i^L if x has a neighbor in X_{i-1}^L and into X_i^R if x has a neighbor in X_{i-1}^R. Note that x has at least one neighbor in X_{i-1}, and so x will be put into at least one of the sets X_i^L or X_i^R. If x is put into both sets X_i^L and X_i^R, the algorithm outputs success with $\hat{u} = x$ and halts. If $|X_i^L| > 2c^2$ or if two vertices in X_i^L have distance at least $2c+1$ from each other in G the algorithm picks a vertex $x \in X_i^L$ and returns success with $\hat{u} = x$. Similarly, if $|X_i^R| > 2c^2$ or if two vertices in X_i^R have distance at least $2c+1$ from each other in G the algorithm picks a vertex $x \in X_i^R$ and returns success with $\hat{u} = x$. If the BFS stops, (i.e $X_i = \emptyset$), the algorithm outputs fail.

4.2.2 STITCHing together approximate line embeddings

We now describe the STITCH algorithm. This algorithm takes as input G, H, c and $F \subseteq V(G)$, runs in polynomial time and halts. We will prove that if there exists a c-embedding f_{OPT} of G into a weighted subdivision $(H_{\mathsf{OPT}}, w_{\mathsf{OPT}})$ of H such that F $2r$-covers all interesting clusters of G, the algorithm produces a c_{ALG}-embedding f_{ALG} of G into a weighted subdivision $(H_{\mathsf{ALG}}, w_{\mathsf{ALG}})$ of a quasi-subgraph H' of H. Throughout this section we will assume that such an embedding f_{OPT} exists.

The STITCH algorithm starts by setting $R = 4r$, $\Delta = 4r$ and then proceeds as follows. As long as there are two vertices u and v in F such that $2R \leq d_G(u,v) \leq 2R + \Delta$, the algorithm increases R to $R + \Delta$. Note that this process will stop after at most $\binom{|F|}{2}$ iterations, and therefore when it terminates we have $R \leq 4r \cdot h^2 \leq 400c^3h^3$. Define $B = \mathsf{ball}_G(F, R)$, and $\mathcal{B}$ to be the family of connected components of $G[B]$. Notice that the previous process ensures that for any $B_1, B_2 \in \mathcal{B}$ we have $d_G(B_1, B_2) \geq \Delta$. Notice further that for every interesting cluster C in H we have that $\mathsf{ball}_G(\hat{f}^{-1}_{\mathsf{OPT}}(C), r) \subseteq B$.

We now classify the connected components of $G - B$. A component Z of $G - B$ is called *deep* if it contains at least one vertex at distance (in G) at least $\frac{\Delta}{2}$ from F, and it is *shallow* otherwise. The shallow components are easy to handle because they only contain vertices close to F.

▶ **Lemma 11.** *For every shallow component Z of $G - B$, there is at most one connected component $B_1 \in \mathcal{B}$ that contains neighbors of Z.*

The next sequence of lemmas allows us to handle deep components. We say that a component Z in $G - B$ *lies on* the cluster-chain $\chi = C_1, e_1, \ldots, C_t$ if

$$Z \subseteq \left(\bigcup_{i \leq t} \hat{f}^{-1}_{\mathsf{OPT}}(C_i) \cup \hat{f}^{-1}_{\mathsf{OPT}}(e_i) \right) \setminus \hat{f}^{-1}_{\mathsf{OPT}}(C_1 \cup C_t).$$

▶ **Lemma 12.** *Every component Z of $G - B$ lies on some cluster-chain and no two deep components Z_1, Z_2 of $G - B$ can lie on the same cluster-chain χ.*

▶ **Lemma 13.** *There is a polynomial time algorithm that given G, B and a component Z of $G - B$ computes an embedding of Z components of $G - B$ into the line with distortion at most $(\ell \cdot h \cdot c)^4$. Furthermore, all vertices in Z with neighbors outside Z are mapped by this embedding within distance $(\ell \cdot h \cdot c)^6$ from the end-points.*

Proof. Let Z be a component of $G - B$. By Lemma 12, Z lies on a cluster-chain $\chi = C_1, e_1, \ldots, C_t$. Define the following total ordering of the vertices in Z: If $\hat{f}_{\mathsf{OPT}}(a) \in C_i \cup \{e_i\}$ and $\hat{f}_{\mathsf{OPT}}(b) \in C_j \cup \{e_j\}$ and $i < j$, then a comes before b. If $\hat{f}_{\mathsf{OPT}}(a) \in C_i$ and $\hat{f}_{\mathsf{OPT}}(b) = e_i$ then a comes before b. If $\hat{f}_{\mathsf{OPT}}(a) = \hat{f}_{\mathsf{OPT}}(b) = e_i$ and $f_{\mathsf{OPT}}(a)$ is closer than $f_{\mathsf{OPT}}(b)$ to C_{i-1}, then a comes before b. If $\hat{f}_{\mathsf{OPT}}(a) \in C_i$ and $\hat{f}_{\mathsf{OPT}}(b) \in C_i$ we place a and b in any relative order.

At most $\ell \cdot h$ vertices are mapped to any boring cluster C_i, and the distance between any two vertices in the same boring cluster in H_{OPT} is at most $\ell \cdot h$. Thus the distance (in G) between any two consecutive vertices in this ordering is at most $\ell \cdot h \cdot c$. The number of vertices appearing in the ordering between the two endpoints of an edge is at most $\ell \cdot h$ (all the vertices of a boring cluster). Thus, if the ordering is turned into a pushing, non-contracting embedding into the line, the distortion of this embedding is at most $(\ell \cdot h)^2 \cdot c$. Using the known polynomial time approximation algorithm for embedding into the line [5] we can find an embedding of Z into the line with distortion at most $(\ell \cdot h \cdot c)^4$ in polynomial time.

Because B is a union of at most h balls, it follows that at most $c^2 \cdot h^2$ vertices in Z have neighbors in G, and that all of these vertices are among the $\ell \cdot h$ first or last ones in the above ordering. Since any two low distortion embeddings of a metric space into the line map the same vertices close to the end-points, it follows that all vertices in Z with neighbors outside Z are mapped by this embedding within distance $(\ell \cdot h \cdot c)^6$ from the end-points. ◀

The STITCH algorithm builds the graph H' as follows. Every vertex of H' corresponds to a connected component $B \in \mathcal{B}$. Every deep component Z of $G - B$ corresponds to an edge between the (at most two) sets B_1 and $B_2 \in \mathcal{B}$ that have non-empty intersection with $N_G(Z)$. Note that the graph H' is a multi-graph because it may have multiple edges and self loops. However, since each set $B \in \mathcal{B}$ has a connected image in H under $\hat{f}$, Lemma 12 implies that H' is a topological subgraph of H. Hence any weighted subdivision of H' is a weighted subdivision of a subgraph of H. The STITCH algorithm uses Lemma 13 to compute embeddings of each deep connected component Z of $G \setminus B$. Further, for each component $B_i \in \mathcal{B}$ the algorithm computes the set $B_i^\star$ which contains B_i, as well as the vertex sets of all shallow connected components whose neighborhood is in B_i. By Lemma 11 the $B_i^\star$'s together with the deep components of $G - B$ form a partition of $V(G)$.

What we would like to do is to map each set $B_i^\star$ onto the vertex of H' that it corresponds to, and map each deep connected component Z of $G - B$ onto the edge of H' that it corresponds to. When mapping Z onto the edge of H we use the computed embedding of Z into the line, and subdivide this edge appropriately.

The reason this does not work directly is that we may not map all the vertices of $B_i^\star$ onto the single vertex v_i in H' that corresponds to B_i. Instead, STITCH picks one of the edges incident to v_i, sub-divides the edge an appropriate number of times, and maps all the vertices of $B_i^\star$ onto the newly created vertices on this edge. The order in which the vertices of $B_i^\star$ are mapped onto the edge is chosen arbitrarily, however all of these vertices are mapped closer to v_i than any vertices of the deep component Z that is mapped onto the edge. This concludes the construction of H_{ALG} and f_{ALG}. The STITCH algorithm defines a weight function w_{ALG} on the edges of H_{ALG}, such that the embedding is pushing and non-contracting.

▶ **Lemma 14.** *f_{ALG} is a c_{ALG}-embedding of G into $(H_{\mathsf{ALG}}, w_{\mathsf{ALG}})$.*

We are now ready to prove Lemma 9.

Proof of Lemma 9. The algorithm runs the algorithm of Lemma 10, to produce a collection $\mathcal{F}$, such that $|\mathcal{F}| \leq 2^h$, every set in $\mathcal{F}$ has size at most h, and such that if G has a c-embedding f_{OPT} of into a weighted subdivision of H, then some $F \in \mathcal{F}$ $2r$-covers all interesting clusters (of f_{OPT}) in G. For each $F \in \mathcal{F}$ the algorithm runs the STITCH algorithm, which takes polynomial time. If STITCH outputs a c_{ALG}-embedding of G into a weighted subdivision of a quasi-subgraph H' of H, the algorithm returns this embedding.

By Lemma 14, for the choice of $F \in \mathcal{F}$ that $2r$-covers all interesting clusters, the STITCH algorithm does output a c_{ALG}-embedding of G into a weighted subdivision of a quasi-subgraph H' of H. This concludes the proof. ◀

The discussion prior to the statement of Lemma 9 immediately implies that Lemma 9 is sufficient to give an approximation algorithm for finding a low distortion (not necessarily pushing, proper or non-contracting) embedding G into a (unweighted) subdivision of H. The only overhead of the algorithm is the guessing of the quasi-subgraph H_{sub} of H, this incurs an additional factor of $3^{|V(H)|+|E(H)|} \leq 9^h$ in the running time, yielding the proof of Theorem 1.

Finally, we remark that at a cost of a potentially higher running time in terms of h, one may replace the $(h+1)^9$ factor with c^9. If $c \geq h+1$ we have that $c_{\mathsf{ALG}} \leq 64 \cdot 10^6 \cdot c^{33}$. On the other hand, if $c \leq h+1$ we may run the algorithm of Theorem 2 in time $f(H)n^{\mathcal{O}(1)}$ instead and solve the problem optimally.

References

1. Sanjeev Arora, James Lee, and Assaf Naor. Euclidean distortion and the sparsest cut. *Journal of the American Mathematical Society*, 21(1):1–21, 2008.
2. Sanjeev Arora, Satish Rao, and Umesh Vazirani. Expander flows, geometric embeddings and graph partitioning. *Journal of the ACM (JACM)*, 56(2):5, 2009.
3. Mihai Bădoiu, Julia Chuzhoy, Piotr Indyk, and Anastasios Sidiropoulos. Low-distortion embeddings of general metrics into the line. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, pages 225–233. ACM, 2005.
4. Mihai Bădoiu, Julia Chuzhoy, Piotr Indyk, and Anastasios Sidiropoulos. Embedding ultrametrics into low-dimensional spaces. In *Proceedings of the 22nd Annual Symposium on Computational Geometry (SoCG)*, pages 187–196. ACM, 2006.
5. Mihai Bădoiu, Kedar Dhamdhere, Anupam Gupta, Yuri Rabinovich, Harald Räcke, R. Ravi, and Anastasios Sidiropoulos. Approximation algorithms for low-distortion embeddings into low-dimensional spaces. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 119–128. SIAM, 2005.
6. Mihai Bădoiu, Piotr Indyk, and Anastasios Sidiropoulos. Approximation algorithms for embedding general metrics into trees. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 512–521. ACM and SIAM, 2007.
7. Yair Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on Foundations of Computer Science*, pages 184–193. IEEE, 1996.
8. Nishanth Chandran, Ryan Moriarty, Rafail Ostrovsky, Omkant Pandey, Mohammad Ali Safari, and Amit Sahai. Improved algorithms for optimal embeddings. *ACM Transactions on Algorithms*, 4(4), 2008. doi:10.1145/1383369.1383376.

9 Mark de Berg, Krzysztof Onak, and Anastasios Sidiropoulos. Fat polygonal partitions with applications to visualization and embeddings. *Journal of Computational Geometry*, 4(1):212–239, 2013.

10 Jeff Edmonds, Anastasios Sidiropoulos, and Anastasios Zouzias. Inapproximability for planar embedding problems. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 222–235. Society for Industrial and Applied Mathematics, 2010.

11 Martin Farach, Sampath Kannan, and Tandy Warnow. A robust model for finding optimal evolutionary trees. *Algorithmica*, 13(1-2):155–179, 1995.

12 Martin Farach-Colton and Piotr Indyk. Approximate nearest neighbor algorithms for hausdorff metrics via embeddings. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 171–179. IEEE, 1999.

13 Michael Fellows, Fedor V. Fomin, Daniel Lokshtanov, Elena Losievskaja, Frances Rosamond, and Saket Saurabh. Distortion is fixed parameter tractable. *ACM Trans. Comput. Theory*, 5(4):16:1–16:20, 2013. doi:10.1145/2489789.

14 Michael R. Fellows, Fedor V. Fomin, Daniel Lokshtanov, Elena Losievskaja, Frances A. Rosamond, and Saket Saurabh. Distortion is fixed parameter tractable. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 5555 of *Lecture Notes in Computer Science*, pages 463–474. Springer, 2009.

15 Alexander Hall and Christos Papadimitriou. Approximating the distortion. In Chandra Chekuri, Klaus Jansen, José D. P. Rolim, and Luca Trevisan, editors, *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 111–122, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

16 Piotr Indyk. Algorithmic applications of low-distortion geometric embeddings. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 10–33. IEEE, 2001.

17 Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *Journal of the ACM (JACM)*, 53(3):307–323, 2006.

18 Piotr Indyk and Jiri Matousek. Low-distortion embeddings of finite metric spaces. In *Handbook of Discrete and Computational Geometry*, pages 177–196. CRC Press, 2004.

19 Claire Kenyon, Yuval Rabani, and Alistair Sinclair. Low distortion maps between point sets. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC)*, pages 272–280. ACM, 2004.

20 Claire Kenyon, Yuval Rabani, and Alistair Sinclair. Low distortion maps between point sets. *SIAM J. Comput.*, 39(4):1617–1636, 2009. doi:10.1137/080712921.

21 Subhash Khot and Rishi Saket. Hardness of embedding metric spaces of equal size. In *Approximation, randomization, and combinatorial optimization. Algorithms and techniques*, pages 218–227. Springer, 2007.

22 Nathan Linial. Finite metric-spaces—combinatorics, geometry and algorithms. In *Proceedings of the International Congress of Mathematicians, Vol. III*, pages 573–586, Beijing, 2002. Higher Ed. Press.

23 Nathan Linial, Eran London, and Yuri Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995.

24 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Slightly superexponential parameterized problems. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 760–776. SIAM, 2011.

25 Jiří Matoušek and Anastasios Sidiropoulos. Inapproximability for metric embeddings into $\mathbb{R}^d$. *Transactions of the American Mathematical Society*, 362(12):6341–6365, 2010.

26 Amir Nayyeri and Benjamin Raichel. Reality distortion: Exact and approximate algorithms for embedding into the line. In *Proceedings of the 56th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 729–747. IEEE, 2015.

27 Amir Nayyeri and Benjamin Raichel. A treehouse with custom windows: Minimum distortion embeddings into bounded treewidth graphs. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '17, pages 724–736, Philadelphia, PA, USA, 2017. Society for Industrial and Applied Mathematics. URL: `http://dl.acm.org/citation.cfm?id=3039686.3039732`.

28 Christos Papadimitriou and Shmuel Safra. The complexity of low-distortion embeddings between point sets. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, volume 5, pages 112–118. SIAM, 2005.

Fast Approximation and Exact Computation of Negative Curvature Parameters of Graphs

Jérémie Chalopin
CNRS, Aix-Marseille Université, Université de Toulon, LIS, Marseille, France
jeremie.chalopin@lis-lab.fr

Victor Chepoi
Aix-Marseille Université, CNRS, Université de Toulon, LIS, Marseille, France
victor.chepoi@lis-lab.fr

Feodor F. Dragan
Computer Science Department, Kent State University, Kent, USA
dragan@cs.kent.edu

Guillaume Ducoffe
National Institute for Research and Development in Informatics and Research Institute of the University of Bucharest, Bucureşti, România
guillaume.ducoffe@ici.ro

Abdulhakeem Mohammed
Computer Science Department, Kent State University, Kent, USA
amohamm4@kent.edu

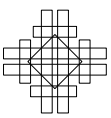
Yann Vaxès
Aix-Marseille Université, CNRS, Université de Toulon, LIS, Marseille, France
yann.vaxes@lis-lab.fr

Abstract

In this paper, we study Gromov hyperbolicity and related parameters, that represent how close (locally) a metric space is to a tree from a metric point of view. The study of Gromov hyperbolicity for geodesic metric spaces can be reduced to the study of *graph hyperbolicity*. Our main contribution in this note is a new characterization of hyperbolicity for graphs (and for complete geodesic metric spaces). This characterization has algorithmic implications in the field of large-scale network analysis, which was one of our initial motivations. A sharp estimate of graph hyperbolicity is useful, e.g., in embedding an undirected graph into hyperbolic space with minimum distortion [Verbeek and Suri, SoCG'14]. The hyperbolicity of a graph can be computed in polynomial-time, however it is unlikely that it can be done in *subcubic* time. This makes this parameter difficult to compute or to approximate on large graphs. Using our new characterization of graph hyperbolicity, we provide a simple factor 8 approximation algorithm for computing the hyperbolicity of an n-vertex graph $G = (V, E)$ in optimal time $O(n^2)$ (assuming that the input is the distance matrix of the graph). This algorithm leads to constant factor approximations of other graph-parameters related to hyperbolicity (thinness, slimness, and insize). We also present the first efficient algorithms for exact computation of these parameters. All of our algorithms can be used to approximate the hyperbolicity of a geodesic metric space.

2012 ACM Subject Classification , Mathematics of computing → Graph algorithms, Mathematics of computing → Approximation algorithms, Theory of computation → Computational geometry

Keywords and phrases Gromov hyperbolicity, Graphs, Geodesic metric spaces, Approximation algorithms



34th International Symposium on Computational Geometry (SoCG 2018).
Editors: Bettina Speckmann and Csaba D. Tóth; Article No. 22; pp. 22:1–22:15

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.22

Related Version A full version [10] of this paper is available at `https://arxiv.org/abs/1803.06324`

Funding The research of J. C., V. C., and Y. V. was supported by ANR project DISTANCIA (ANR-17-CE40-0015).

Acknowledgements We are grateful to the referees of this short version for useful comments.

1 Introduction

Understanding the geometric properties of complex networks is a key issue in network analysis and geometry of graphs. One important such property is the *negative curvature* [29], causing the traffic between the vertices to pass through a relatively small core of the network – as if the shortest paths between them were curved inwards. It has been empirically observed, then formally proved [14], that such a phenomenon is related to the value of the *Gromov hyperbolicity* of the graph. In this paper, we propose exact and approximation algorithms to compute hyperbolicity of a graph and its relatives (the approximation algorithms can be applied to geodesic metric spaces as well).

A metric space (X, d) is δ-*hyperbolic* [3, 9, 26] if for any four points w, v, x, y of X, the two largest of the distance sums $d(w, v) + d(x, y)$, $d(w, x) + d(v, y)$, $d(w, y) + d(v, x)$ differ by at most $2\delta \geq 0$. A graph $G = (V, E)$ endowed with its standard graph-distance d_G is δ-*hyperbolic* if the metric space (X, d_G) is δ-hyperbolic. In case of geodesic metric spaces and graphs, δ-hyperbolicity can be defined in other equivalent ways, e.g., via thin or slim geodesic triangles. The hyperbolicity $\delta(X)$ of a metric space X is the smallest $\delta \geq 0$ such that X is δ-hyperbolic. The hyperbolicity $\delta(X)$ can be viewed as a local measure of how close X is to a tree: the smaller the hyperbolicity is, the closer the metrics of its 4-point subspaces are close to tree-metrics.

The study of hyperbolicity of graphs is motivated by the fact that many real-world graphs are tree-like from a metric point of view [1, 2, 6] or have small hyperbolicity [28, 29, 32]. This is due to the fact that many of these graphs (including Internet application networks, web networks, collaboration networks, social networks, biological networks, and others) possess certain geometric and topological characteristics. Hence, for many applications, including the design of efficient algorithms (cf., e.g., [6, 11–15, 19, 22, 34]), it is useful to know the hyperbolicity $\delta(G)$ of a graph G.

Related work. For an n-vertex graph G, the definition of hyperbolicity directly implies a simple brute-force $O(n^4)$ algorithm to compute $\delta(G)$. This running time is too slow for computing the hyperbolicity of large graphs that occur in applications [1, 6, 7, 24]. On the theoretical side, it was shown that relying on matrix multiplication results, one can improve the upper bound on time-complexity to $O(n^{3.69})$ [24]. Moreover, roughly quadratic lower bounds are known [7, 17, 24]. In practice, however, the best known algorithm still has an $O(n^4)$-time worst-case bound but uses several clever tricks when compared to the brute-force algorithm [6]. Based on empirical studies, an $O(mn)$ running time is claimed, where m is the number of edges in the graph. Furthermore, there are heuristics for computing the hyperbolicity of a given graph [16], and there are investigations whether one can compute hyperbolicity in linear time when some graph parameters take small values [18, 23].

Perhaps it is interesting to notice that the first algorithms for testing graph hyperbolicity were designed for Cayley graphs of finitely generated groups (these are infinite vertex-transitive graphs of uniformly bounded degrees). Gromov gave an algorithm to recognize Cayley graphs of hyperbolic groups and estimate the hyperbolicity constant δ. His algorithm is based on the theorem that in Cayley graphs, the hyperbolicity "propagates", i.e., if balls of an appropriate fixed radius induce a δ-hyperbolic space, then the whole space is δ'-hyperbolic for some $\delta' > \delta$ (see [26], 6.6.F and [20]). Therefore, in order to check the hyperbolicity of a Cayley graph, it is enough to verify the hyperbolicity of a sufficiently big ball (all balls of a given radius in a Cayley graph are isomorphic to each other). For other algorithms deciding if the Cayley graph of a finitely generated group is hyperbolic, see [8, 30]. However, similar methods do not help when dealing with arbitrary graphs.

By a result of Gromov [26], if the four-point condition in the definition of hyperbolicity holds for a fixed basepoint w and any triplet x, y, v of X, then the metric space (X, d) is 2δ-hyperbolic. This provides a factor 2 approximation of hyperbolicity of a metric space on n points running in cubic $O(n^3)$ time. Using fast algorithms for computing (max,min)-matrix products, it was noticed in [24] that this 2-approximation of hyperbolicity can be implemented in $O(n^{2.69})$ time. In the same paper, it was shown that any algorithm computing the hyperbolicity for a fixed basepoint in time $O(n^{2.05})$ would provide an algorithm for $(\max, \min)$-matrix multiplication faster than the existing ones. In [21], approximation algorithms are given to compute a $(1+\epsilon)$-approximation in $O(\epsilon^{-1}n^{3.38})$ time and a $(2+\epsilon)$-approximation in $O(\epsilon^{-1}n^{2.38})$ time. As a direct application of the characterization of hyperbolicity of graphs via a cop and robber game and dismantlability, [11] presents a simple constant factor approximation algorithm for hyperbolicity of G running in optimal $O(n^2)$ time. Its approximation ratio is huge (1569), however it is believed that its theoretical performance is much better and the factor of 1569 is mainly due to the use in the proof of the definition of hyperbolicity via linear isoperimetric inequality. This shows that the question of designing fast and (theoretically certified) accurate algorithms for approximating graph hyperbolicity is still an important and open question.

Our contribution. In this paper, we tackle this open question and propose a very simple (and thus practical) factor 8 algorithm for approximating the hyperbolicity $\delta(G)$ of an n-vertex graph G running in optimal $O(n^2)$ time. As in several previous algorithms, we assume that the input is the distance matrix D of the graph G. Our algorithm picks a basepoint w, a Breadth-First-Search tree T rooted at w, and considers only geodesic triangles of G with one vertex at w and two sides on T. For all such sides in T, it computes the maximum over all distances between the two preimages of the centers of the respective tripods. This maximum $\rho_{w,T}$ can be easily computed in $O(n^2)$ time and provides an 8-approximation for $\delta(G)$. For complete geodesic spaces (X, d), we show that we can always define a geodesic spanning tree T based at any point w of X, and that the same relationships between $\rho_{w,T}$ and the hyperbolicity of (X, d) hold, thus providing a new characterization of hyperbolicity. Perhaps it is surprising that hyperbolicity that is originally defined via quadruplets and can be 2-approximated via triplets (i.e., via pointed hyperbolicity), can be finally defined and approximated only via pairs (and an arbitrary fixed BFS-tree). We hope that this new characterization can be useful in establishing that graphs and simplicial complexes occurring in geometry and in network analysis are hyperbolic.

The way $\rho_{w,T}$ is computed is closely related to how hyperbolicity is defined via slimness, thinness, and insize of its geodesic triangles. Similarly to the hyperbolicity $\delta(G)$, one can define slimness $\varsigma(G)$, thinness $\tau(G)$, and insize $\iota(G)$ of a graph G. As a direct consequence of our

algorithm for approximating $\delta(G)$ and the relationships between $\delta(G)$ and $\varsigma(G), \tau(G), \iota(G)$, we obtain constant factor $O(n^2)$ time algorithms for approximating these parameters. On the other hand, an *exact* computation, in polynomial time, of these geometric parameters has long stayed elusive. This is due to the fact that $\varsigma(G), \tau(G), \iota(G)$ are defined as minima of some functions over all the geodesic triangles of G, and that there may be exponentially many such triangles. In this paper we provide the first polynomial time algorithms for computing $\varsigma(G), \tau(G)$, and $\iota(G)$. Namely, we show that the thinness $\tau(G)$ and the insize $\iota(G)$ of G can be computed in $O(n^2m)$ time and the slimness $\varsigma(G)$ of G can be computed in $\widehat{O}(n^2m + n^4/\log^3 n)$ time[1]. However, we show that the minimum value of of $\rho_{w,T}$ over all basepoints w and all BFS-trees T cannot be approximated with a factor strictly better than 2 unless P = NP.

2 Gromov hyperbolicity and its relatives

2.1 Gromov hyperbolicity

Let (X, d) be a metric space and $w \in X$. The *Gromov product*[2] of $y, z \in X$ with respect to w is $(y|z)_w = \frac{1}{2}(d(y, w) + d(z, w) - d(y, z))$. A metric space (X, d) is *δ-hyperbolic* [26] for $\delta \geq 0$ if $(x|y)_w \geq \min\{(x|z)_w, (y|z)_w\} - \delta$ for all $w, x, y, z \in X$. Equivalently, (X, d) is δ-hyperbolic if for any $u, v, x, y \in X$, the two largest of the sums $d(u, v) + d(x, y)$, $d(u, x) + d(v, y)$, $d(u, y) + d(v, x)$ differ by at most $2\delta \geq 0$. A metric space (X, d) is said to be *δ-hyperbolic with respect to a basepoint w* if $(x|y)_w \geq \min\{(x|z)_w, (y|z)_w\} - \delta$ for all $x, y, z \in X$.

▶ **Proposition 1** ([3, 9, 25, 26]). *If (X, d) is δ-hyperbolic with respect to some basepoint, then (X, d) is 2δ-hyperbolic.*

Let (X, d) be a metric space. An *(x, y)-geodesic* is a (continuous) map $\gamma : [0, d(x, y)] \to X$ from the segment $[0, d(x, y)]$ of $\mathbb{R}^1$ to X such that $\gamma(0) = x, \gamma(d(x, y)) = y$, and $d(\gamma(s), \gamma(t)) = |s - t|$ for all $s, t \in [0, d(x, y)]$. A *geodesic segment* with endpoints x and y is the image of the map γ (when it is clear from the context, by a geodesic we mean a geodesic segment and we denote it by $[x, y]$). A metric space (X, d) is *geodesic* if every pair of points in X can be joined by a geodesic. A *real tree* (or an *$\mathbb{R}$-tree*) [9, p.186] is a geodesic metric space (T, d) such that

(1) there is a unique geodesic $[x, y]$ joining each pair of points $x, y \in T$;
(2) if $[y, x] \cap [x, z] = \{x\}$, then $[y, x] \cup [x, z] = [y, z]$.

Let (X, d) be a geodesic metric space. A *geodesic triangle* $\Delta(x, y, z)$ with $x, y, z \in X$ is the union $[x, y] \cup [x, z] \cup [y, z]$ of three geodesics connecting these points. A geodesic triangle $\Delta(x, y, z)$ is called *δ-slim* if for any point u on the side $[x, y]$ the distance from u to $[x, z] \cup [z, y]$ is at most δ. Let m_x be the point of $[y, z]$ located at distance $\alpha_y := (x|z)_y$ from y. Then, m_x is located at distance $\alpha_z := (y|x)_z$ from z because $\alpha_y + \alpha_z = d(y, z)$. Analogously, define the points $m_y \in [x, z]$ and $m_z \in [x, y]$ both located at distance $\alpha_x := (y|z)_x$ from x; see Fig. 1 for an illustration. We define a tripod $T(x, y, z)$ consisting of three solid segments $[x, m], [y, m]$, and $[z, m]$ of lengths α_x, α_y, and α_z, respectively. The function mapping the vertices x, y, z of $\Delta(x, y, z)$ to the respective leaves of $T(x, y, z)$ extends uniquely to a function $\varphi : \Delta(x, y, z) \to T(x, y, z)$ such that the restriction of φ on each side of $\Delta(x, y, z)$ is an isometry. This function maps the points m_x, m_y, and m_z to the center m of $T(x, y, z)$. Any

[1] The $\widehat{O}(\cdot)$ notation hides polyloglog factors.
[2] Informally, $(y|z)_w$ can be viewed as half the detour you make, when going over w to get from y to z.

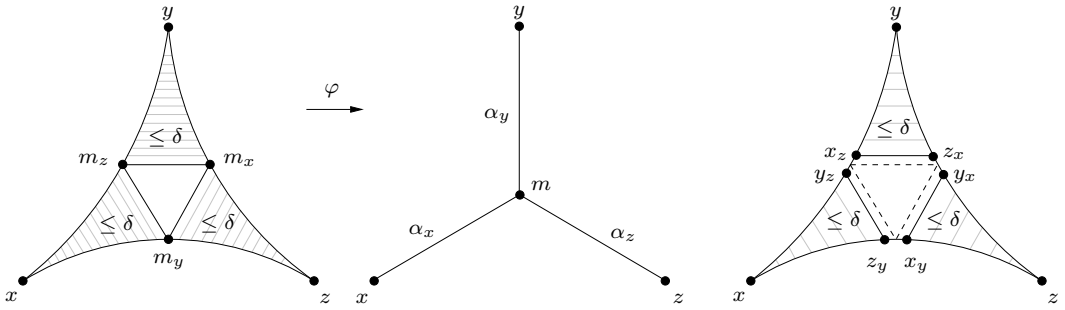


Figure 1 Insize and thinness in geodesic spaces and graphs.

other point of $T(x,y,z)$ is the image of exactly two points of $\Delta(x,y,z)$. A geodesic triangle $\Delta(x,y,z)$ is called δ-*thin* if for all points $u,v \in \Delta(x,y,z)$, $\varphi(u)=\varphi(v)$ implies $d(u,v)\leq\delta$. The *insize* of $\Delta(x,y,z)$ is the diameter of the preimage $\{m_x,m_y,m_z\}$ of the center m of the tripod $T(x,y,z)$. Below, we remind that the hyperbolicity of a geodesic space can be approximated by the maximum thinness and slimness of its geodesic triangles.

For a geodesic metric space (X,d), one can define the following parameters:

- *hyperbolicity* $\delta(X)=\min\{\delta : X \text{ is } \delta\text{-hyperbolic}\}$,
- *pointed hyperbolicity* $\delta_w(X)=\min\{\delta : X \text{ is } \delta\text{-hyperbolic with respect to a basepoint } w\}$,
- *slimness* $\varsigma(X)=\min\{\delta : \text{any geodesic triangle of } X \text{ is } \delta\text{-slim}\}$,
- *thinness* $\tau(X)=\min\{\delta : \text{any geodesic triangle of } X \text{ is } \delta\text{-thin}\}$,
- *insize* $\iota(X)=\min\{\delta : \text{the insize of any geodesic triangle of } X \text{ is at most } \delta\}$.

▶ **Proposition 2** ([3,9,25,26,33]). *For a geodesic metric space* (X,d), $\delta(X)\leq\iota(X)=\tau(X)\leq 4\delta(X)$, $\varsigma(X)\leq\tau(X)\leq 4\varsigma(X)$, *and* $\delta(X)\leq 2\varsigma(X)\leq 3\delta(X)$.

Due to Propositions 1 and 2, a geodesic metric space (X,d) is called *hyperbolic* if one of the numbers $\delta(X),\delta_w(X),\varsigma(X),\tau(X),\iota(X)$ (and thus all) is finite. Notice also that a geodesic metric space (X,d) is 0-hyperbolic if and only if (X,d) is a real tree [9, p.399] (and in this case, $\varsigma(X)=\tau(X)=\iota(X)=\delta(X)=0$).

2.2 Hyperbolicity of graphs

All graphs $G=(V,E)$ occurring in this paper are undirected and connected, but not necessarily finite (in algorithmic results they will be supposed to be finite). For any two vertices $x,y\in V$, the *distance* $d(x,y)$ is the minimum number of edges in a path between x and y. Let $[x,y]$ denote a shortest path connecting vertices x and y in G; we call $[x,y]$ a *geodesic* between x and y. The *interval* $I(u,v)=\{x\in V : d(u,x)+d(x,v)=d(u,v)\}$ consists of all vertices on (u,v)-geodesics. There is a strong analogy between the metric properties of graphs and geodesic metric spaces, due to their uniform local structure. Any graph $G=(V,E)$ gives rise to a geodesic space (X_G,d) (into which G isometrically embeds) obtained by replacing each edge xy of G by a segment isometric to $[0,1]$ with ends at x and y. X_G is called a *metric graph*. Conversely, by [9, Proposition 8.45], any geodesic metric space (X,d) is (3,1)-quasi-isometric to a graph $G=(V,E)$. This graph G is constructed in the following way: let V be an open maximal $\frac{1}{3}$-packing of X, i.e., $d(x,y)>\frac{1}{3}$ for any $x,y\in V$ (that exists by Zorn's lemma). Then two points $x,y\in V$ are adjacent in G if and only if $d(x,y)\leq 1$. Since hyperbolicity is preserved (up to a constant factor) by quasi-isometries, this reduces the computation of hyperbolicity for geodesic spaces to the case of graphs.

The notions of geodesic triangles, insize, δ-slim and δ-thin triangles can also be defined in case of graphs with the single difference that for graphs, the center of the tripod is not necessarily the image of any vertex on the sides of $\Delta(x,y,z)$. For graphs, we "discretize" the notion of δ-thin triangles in the following way. We say that a geodesic triangle $\Delta(x,y,z)$ of a graph G is *δ-thin* if for any $v \in \{x,y,z\}$ and vertices $a \in [v,u]$ and $b \in [v,w]$ ($u,w \in \{x,y,z\}$, and u,v,w are pairwise different), $d(v,a) = d(v,b) \leq (u|w)_v$ implies $d(a,b) \leq \delta$. A graph G is *δ-thin*, if all geodesic triangles in G are δ-thin. Given a geodesic triangle $\Delta(x,y,z) := [x,y] \cup [x,z] \cup [y,z]$ in G, let x_y and y_x be the vertices of $[z,x]$ and $[z,y]$, respectively, both at distance $\lfloor (x|y)_z \rfloor$ from z. Similarly, one can define vertices x_z, z_x and vertices y_z, z_y; see Fig. 1. The *insize* of $\Delta(x,y,z)$ is defined as $\max\{d(y_z,z_y), d(x_y,y_x), d(x_z,z_x)\}$. An interval $I(x,y)$ is said to be κ-thin if $d(a,b) \leq \kappa$ for all $a,b \in I(x,y)$ with $d(x,a) = d(x,b)$. The smallest κ for which all intervals of G are κ-thin is called the *interval thinness* of G and denoted by $\kappa(G)$. Denote also by $\delta(G)$, $\delta_w(G)$, $\varsigma(G)$, $\tau(G)$, and $\iota(G)$ respectively the hyperbolicity, the pointed hyperbolicity with respect to a basepoint w, the slimness, the thinness, and the insize of a graph G. We will need the following inequalities between $\varsigma(G)$, $\tau(G)$, $\iota(G)$, and $\delta(G)$. They are known to be true for all geodesic spaces (see [3, 9, 25, 26, 33]):

▶ **Proposition 3.** *$\delta(G) - \frac{1}{2} \leq \iota(G) = \tau(G) \leq 4\delta(G)$, $\varsigma(G) \leq \tau(G) \leq 4\varsigma(G)$, $\delta(G) - \frac{1}{2} \leq 2\varsigma(G) \leq 6\delta(G) + 1$, and $\kappa(G) \leq \min\{\tau(G), 2\delta(G), 2\varsigma(G)\}$.*

3 Geodesic spanning trees

In this section, we outline the proof that any complete geodesic metric space (X,d) has a geodesic spanning tree rooted at any basepoint w. We hope that this general result will be useful in other contexts. For graphs this is well-known and simple, and such trees can be constructed in various ways, for example via Breadth-First-Search. The existence of BFS-trees in infinite graphs has been established by Polat [31]. However for complete geodesic spaces this result seems to be new (and not completely trivial) and we consider it as one of the main results of the paper. A *geodesic spanning tree rooted at a point w* (a *GS-tree* for short) of a geodesic space (X,d) is a union of geodesics $\Gamma_w := \bigcup_{x \in X} \gamma_{w,x}$ with one end at w such that $y \in \gamma_{w,x}$ implies that $\gamma_{w,y} \subseteq \gamma_{w,x}$. Finally recall that a metric space (X,d) is called *complete* if every Cauchy sequence of points in (X,d) has a limit in X.

▶ **Theorem 4.** *For any complete geodesic metric space (X,d) and for any basepoint w one can define a geodesic spanning tree $\Gamma_w = \bigcup_{x \in X} \gamma_{w,x}$ rooted at w and a real tree $T = (X, d_T)$ such that any $\gamma_{w,x} \in \Gamma_w$ is the unique (w,x)-geodesic of T.*

The first assertion of the theorem immediately follows from the following proposition:

▶ **Proposition 5.** *For any complete geodesic metric space (X,d), for any pair of points $x,y \in X$ one can define an (x,y)-geodesic $\gamma_{x,y}$ such that for all $x,y \in X$ and for all $u,v \in \gamma_{x,y}$, we have $\gamma_{u,v} \subseteq \gamma_{x,y}$.*

Proof. Let $\preceq$ be a well-order on X. For any $x,y \in X$ we define inductively two sets $P^{\prec v}_{x,y}$ and $P^{v}_{x,y}$ for any $v \in X$:

$$P^{\prec v}_{x,y} = \{x,y\} \cup \bigcup_{u \prec v} P^{u}_{x,y},$$

$$P^{v}_{x,y} = \begin{cases} P^{\prec v}_{x,y} \cup \{v\} & \text{if there is an } (x,y)\text{-geodesic } \gamma \text{ with } P^{\prec v}_{x,y} \cup \{v\} \subseteq \gamma, \\ P^{\prec v}_{x,y} & \text{otherwise.} \end{cases}$$

We set $P_{x,y} = \bigcup_{u \in X} P^u_{x,y}$. Using transfinite induction, we prove that there exists an (x,y)-geodesic $\gamma^{\prec v}_{x,y}$ (respectively, $\gamma^v_{x,y}$, $\gamma_{x,y}$) containing $P^{\prec v}_{x,y}$ (respectively, $P^v_{x,y}$, $P_{x,y}$), and we show that $P_{x,y}$ is an (x,y)-geodesic. By the definition of $P_{x,y}$ and $P_{x,u}$, we can show that $P_{x,u} = P_{x,y} \cap B(x, d(x,u))$ for any $u \in P_{x,y}$, and it follows that $P_{u,v} \subseteq P_{x,v} \subseteq P_{x,y}$ for any $u, v \in P_{x,y}$ such that $d(x,u) \leq d(x,v)$. ◀

Consequently, $\Gamma_w = \bigcup_{x \in X} \gamma_{w,x}$ is a geodesic spanning tree of (X,d) rooted at w. Using Γ_w, we can define a real tree T as follows. For any $x \in X$, denote by $[w,x]$ the geodesic segment between w and x which is the image of the geodesic map $\gamma_{w,x}$. From the definition of Γ_w, if $x' \in [w,x]$, then $[w,x'] \subseteq [w,x]$. From the continuity of geodesic maps and the definition of Γ_w it follows that for any two geodesics $\gamma_{w,x}, \gamma_{w,y} \in \Gamma_w$ the intersection $[w,x] \cap [w,y]$ is the image $[w,z]$ of some geodesic $\gamma_{w,z} \in \Gamma_w$. Call z the *lowest common ancestor* of x and y (with respect to the root w) and denote it by $\mathrm{lca}(x,y)$. Define d_T by setting $d_T(w,x) := d(w,x)$ and $d_T(x,y) := d(w,x) + d(w,y) - 2d(w,z) = d(x,z) + d(z,y)$ for any two points $x, y \in X$. We prove that $T = (X, d_T)$ is a real tree. To do so, we show in particular that for any $x, y \in X$, $[x,z] \cup [z,y]$ is the unique (x,y)-geodesic in T where $z = \mathrm{lca}(x,y)$, $[x,z]$ is the portion of the geodesic segment $[w,x]$ between x and z, and $[z,y]$ is the portion of the geodesic segment $[w,y]$ between z and y.

4 Fast approximation

In this section, we introduce a new parameter ρ of a graph G (or of a geodesic space X). This parameter depends on an arbitrary fixed BFS-tree of G (or a GS-tree of X). It can be computed efficiently and it provides constant-factor approximations for $\delta(G)$, $\varsigma(G)$, and $\tau(G)$. In particular, we obtain a very simple factor 8 approximation algorithm for the hyperbolicity $\delta(G)$ of an n-vertex graph G running in optimal $O(n^2)$ time (assuming that the input is the distance matrix of G).

4.1 Fast approximation of hyperbolicity

Consider a graph $G = (V,E)$ or a complete geodesic space (X,d) and an arbitrary BFS-tree T or GS-tree T, respectively, rooted at some vertex or point w (see Section 3). Denote by x_y the point of $[w,x]_T$ at distance $\lfloor (x|y)_w \rfloor$ (resp., $(x|y)_w$) from w and by y_x the point of $[w,y]_T$ at distance $\lfloor (x|y)_w \rfloor$ (resp., $(x|y)_w$) from w. In case of graphs, x_y and y_x are vertices of G. Let $\rho_{w,T} := \sup\{d(x_y, y_x) : x, y \in X\}$. In some sense, $\rho_{w,T}$ can be seen as the insize of G with respect to w and T: the differences between $\rho_{w,T}$ and $\iota(G)$ are that we consider only geodesic triangles $\Delta(w,x,y)$ containing w where the geodesics $[w,x]$ and $[w,y]$ belong to T, and we consider only $d(x_y, y_x)$, instead of $\max\{d(x_y,y_x), d(x_w,w_x), d(y_w,w_y)\}$. Using T, we can also define the thinness of G with respect to w and T: let $\mu_{w,T} = \sup\{d(x',y') : \exists x, y$ such that $x' \in [w,x]_T, y' \in [w,y]_T$ and $d(w,x') = d(w,y') \leq (x|y)_w\}$. Using the same ideas as in the proofs of Propositions 2 and 3 establishing that $\iota(X) = \tau(X)$ and $\iota(G) = \tau(G)$, we can show that these two definitions give rise to the same value.

▶ **Proposition 6.** *For any geodesic space X and any GS-tree T rooted at a point w, $\rho_{w,T} = \mu_{w,T}$. Analogously, for any graph G and any BFS-tree T rooted at w, $\rho_{w,T} = \mu_{w,T}$.*

In the following, when w and T are clear from the context, we denote $\rho_{w,T}$ by ρ. The next theorem is the main result of this paper. It establishes that 2ρ provides an 8-approximation of the hyperbolicity of $\delta(G)$ or $\delta(X)$, and that in the case of a finite graph G, ρ can be computed in $O(n^2)$ time when the distance matrix D of G is given.

▶ **Theorem 7.** *Given a graph G (respectively, a geodesic space X) and a BFS-tree T (respectively, a GS-tree T) rooted at w,*

(1) *$\delta(G) \leq 2\rho_{w,T} + 1 \leq 8\delta(G) + 1$ (respectively, $\delta(X) \leq 2\rho_{w,T} \leq 8\delta(X)$).*

(2) *If G has n vertices, given the distance matrix D of G, $\rho_{w,T}$ can be computed in $O(n^2)$ time. Consequently, an 8-approximation (with an additive constant 1) of the hyperbolicity $\delta(G)$ of G can be found in $O(n^2)$ time.*

Proof. We prove the first assertion of the theorem for graphs (for geodesic spaces, the proof is similar). Let $\rho := \rho_{w,T}$, $\delta := \delta(G)$ and $\delta_w := \delta_w(G)$. By Gromov's Proposition 1, $\delta \leq 2\delta_w$. We proceed in two steps. In the first step, we show that $\rho \leq 4\delta$. In the second step, we prove that $\delta_w \leq \rho + \frac{1}{2}$. Hence, combining both steps we obtain $\delta \leq 2\delta_w \leq 2\rho + 1 \leq 8\delta + 1$.

The first assertion follows from Proposition 3 and from the inequality $\rho \leq \iota(G) = \tau(G)$. To prove that $\delta_w \leq \rho + \frac{1}{2}$, for any quadruplet x, y, z, w containing w, we show the four-point condition $d(x,z) + d(y,w) \leq \max\{d(x,y) + d(z,w), d(y,z) + d(x,w)\} + (2\rho + 1)$. Assume without loss of generality that $d(x,z) + d(y,w) \geq \max\{d(x,y) + d(z,w), d(y,z) + d(x,w)\}$ and that $d(w,x_y) = d(w,y_x) \leq d(w,y_z) = d(w,z_y)$. From the definition of ρ, $d(x_y,y_x) \leq \rho$ and $d(y_z,z_y) \leq \rho$. Consequently, by the definition of x_y, y_x, y_z, z_y and by the triange inequality, we get

$$\begin{aligned} d(y,w) + d(x,z) &\leq d(y,w) + d(x,x_y) + d(x_y,y_x) + d(y_x,y_z) + d(y_z,z_y) + d(z_y,z) \\ &\leq (d(y,y_z) + d(y_z,w)) + d(x,x_y) + \rho + d(y_x,y_z) + \rho + d(z_y,z) \\ &= d(y,y_z) + d(w,z_y) + d(x,x_y) + d(y_x,y_z) + d(z_y,z) + 2\rho \\ &= d(y,y_z) + d(x,x_y) + (d(y,y_x) - d(y,y_z)) + d(w,z) + 2\rho \\ &\leq d(x,y) + 1 + d(w,z) + 2\rho, \end{aligned}$$

the last inequality following from the definition of x_y and y_x in graphs (in the case of geodesic metric spaces, we have $d(x,x_y) + d(y,y_x) = d(x,y)$). This establishes the four-point condition for w, x, y, z, and proves that $\delta_w \leq \rho + \frac{1}{2}$.

We present now a simple self-contained algorithm for computing ρ in $O(n^2)$ time when $G = (V,E)$ is a graph with n vertices. (Its space complexity can be improved using the algorithm of [4,5] for computing level d ancestors in trees.) For any non-negative integer r, let $x(r)$ be the unique vertex of $[w,x]_T$ at distance r from w if $r < d(w,x)$ and the vertex x if $r \geq d(w,x)$. First, we compute in $O(n^2)$ time a table M with lines indexed by V, columns indexed by $\{1,\ldots,n\}$, and such that $M(x,r)$ is the identifier of the vertex $x(r)$ of $[w,x]_T$ located at distance r from w. To compute this table, we explore the tree T starting from w. Let x be the current vertex and r its distance to the root w. For every vertex y in the subtree of T rooted at x, we set $M(y,r) := x$. Assuming that the table M and the distance matrix $D := (d(u,v) : u,v \in X)$ between the vertices of G are available, we can compute $x_y = M(x, \lfloor (x|y)_w \rfloor)$, $y_x = M(y, \lfloor (x|y)_w \rfloor)$ and $d(x_y,y_x)$ in constant time for each pair of vertices x, y, and thus $\rho = \max\{d(x_y,y_x) : x,y \in V\}$ can be computed in $O(n^2)$ time. ◀

Theorem 7 provides a new characterization of infinite hyperbolic graphs.

▶ **Corollary 8.** *Consider an infinite graph G and an arbitrary BFS-tree T rooted at a vertex w. The graph G is hyperbolic if and only if $\rho_{w,T} < \infty$.*

The following result shows that the bounds in Theorem 7 are optimal.

▶ **Proposition 9.** *For any positive integer k, there exists a graph H_k, a vertex w, and a BFS-tree T rooted at w such that $\delta(H_k) = k$ and $\rho_{w,T} = 4k$.*

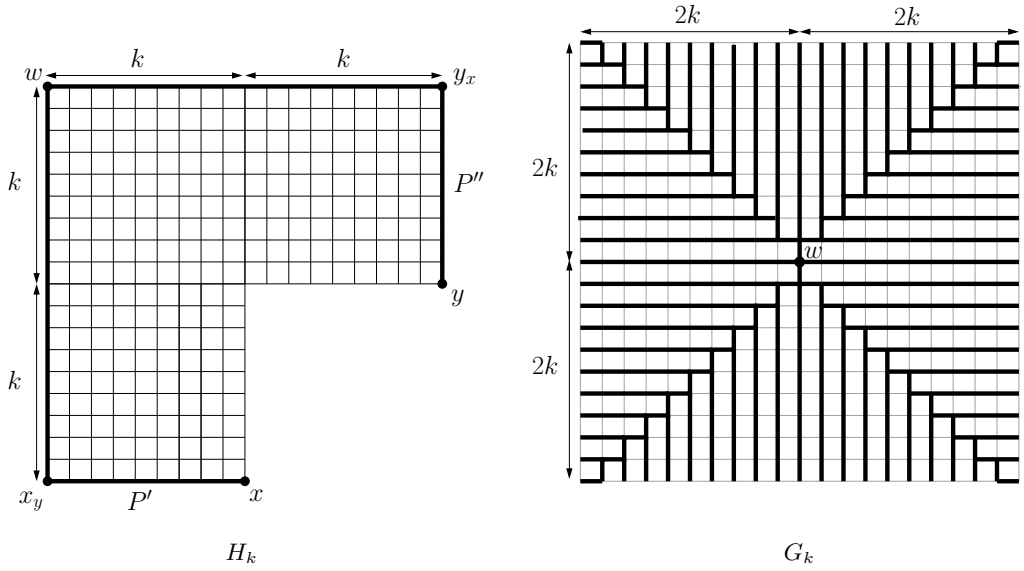


Figure 2 In H_k, $\rho_{w,T} = d(x_y, y_x) = 4\delta(H_k)$, showing that the inequality $\rho \leq 4\delta$ is tight in the proof of Theorem 7. In G_k, $\rho_{w,T} \leq 2k = \frac{1}{2}\delta(G_k)$, showing that (up to an additive factor of 1) the inequality $\delta \leq 2\rho + 1$ is tight in the proof of Theorem 7.

For any positive integer k, there exists a graph G_k, a vertex w, and a BFS-tree T rooted at w such that $\rho_{w,T} \leq 2k$ and $\delta(G_k) = 4k$.

Proof. The graph H_k is the $2k \times 2k$ square grid from which we removed the vertices of the rightmost and downmost $(k-1) \times (k-1)$ square (see Fig. 2, left). The graph H_k is a median graph and therefore its hyperbolicity is the size of a largest isometrically embedded square subgrid [12, 27]. The largest square subgrid of H_k has size k, thus $\delta(H_k) = k$.

Let w be the leftmost upmost vertex of H_k. Let x be the downmost rightmost vertex of H_k and y be the rightmost downmost vertex of H_k. Then $d(x, y) = 2k$ and $d(x, w) = d(y, w) = 3k$. Let P' and P'' be the shortest paths between w and x and w and y, respectively, running on the boundary of H_k. Let T be any BFS-tree rooted at w and containing the shortest paths P' and P''. The vertices $x_y \in P'$ and $y_x \in P''$ are located at distance $(x|y)_w = 2k$ from w. Thus x_y is the leftmost downmost vertex and y_x is the rightmost upmost vertex. Hence $\rho_{w,T} \geq d(x_y, y_x) = 4k$. Since the diameter of H_k is $4k$, we conclude that $\rho_{w,T} = 4k = 4\delta(H_k)$.

Let G_k be the $4k \times 4k$ square grid and note that $\delta(G_k) = 4k$. Let w be the center of G_k. Note that $\delta(G_k) = 4k$. We suppose that G_k is isometrically embedded in the ℓ_1-plane in such a way that w is mapped to the origin of coordinates $(0, 0)$ and the four corners of G_k are mapped to the points with coordinates $(2k, 2k), (-2k, 2k), (-2k, -2k), (2k, -2k)$, We build the BFS-tree T of G_k as follows. First we connect w to each of the corners of G_k by a shortest zigzagging path (see Fig. 2, right). For each $i \leq i \leq k$, we add a vertical path from (i, i) to $(i, 2k)$, from $(i, -i)$ to $(i, -2k)$, from $(-i, i)$ to $(-i, 2k)$, and from $(-i, -i)$ to $(-i, -2k)$. Similarly, for each $i \leq i \leq k$, we add a horizontal path from (i, i) to $(2k, i)$, from $(i, -i)$ to $(2k, -i)$, from $(-i, i)$ to $(-2k, i)$, and from $(-i, -i)$ to $(-2k, -i)$. For any vertex v, the shortest path of G_k connecting w to v in T has the following structure: it starts by a subpath of one of the zigzagging paths until it reaches the vertical or horizontal line containing v and then it continues along this line until v. One can show that $\rho_{w,T} \leq 2k = \frac{1}{2}\delta(G_k)$. ◀

The definition of $\rho_{w,T}$ depends on the choice of the basepoint w and of the BFS-tree T rooted at w. We show below that the best choices of w and T do not improve the bounds in Theorem 7. For a graph G, let $\rho_-(G) = \min\{\rho_{w,T} : w \in V \text{ and } T \text{ is a BFS-tree rooted at } w\}$ and call $\rho_-(G)$ the *minsize* of G. On the other hand, the *maxsize* $\rho_+(G) = \max\{\rho_{w,T} : w \in V \text{ and } T \text{ is a BFS-tree rooted at } w\}$ of G coincides with its insize $\iota(G)$. Indeed, from the definition, $\rho_+(G) \leq \iota(G)$. Conversely, consider a geodesic triangle $\Delta(x, y, w)$ maximizing the insize and suppose, without loss of generality, that $d(x_y, y_x) = \iota(G)$, where x_y and y_x are chosen on the sides of $\Delta(x, y, w)$. Then, if we choose a BFS-tree rooted at w, and such that x_y is an ancestor of x and y_x is an ancestor of y, then one obtains that $\rho_+(G) \geq \iota(G)$. We show in Section 5 that $\rho_+(G) = \tau(G)$ can be computed in polynomial time, and by Proposition 3, it gives a 4-approximation of $\delta(G)$.

On the other hand, the next proposition shows that one cannot get better than a factor 8 approximation of hyperbolicity if instead of computing $\rho_{w,T}$ for an arbitrary BFS tree T rooted at some arbitrary vertex w, we compute the minsize $\rho_-(G)$. Furthermore, we show in Section 5 that we cannot approximate $\rho_-(G)$ with a factor strictly better than 2 unless P = NP. In order to prove the following proposition, we modify slightly the graph H_k of Proposition 9 so that the shortest paths from w to x and y become unique, and then we glue two copies of this modified graph in w.

▶ **Proposition 10.** *For any positive integer k, there exists a graph H_k^* with $\delta(H_k^*) = k + O(1)$ and $\rho_+(H_k^*) \geq \rho_-(H_k^*) \geq 4k - 2$ and a graph G_k^* with $\delta(G_k^*) = 4k$ and $\rho_-(G_k^*) \leq 2k$.*

If instead of knowing the distance-matrix D, we only know the distances between the vertices of G up to an additive error k, then we can define a parameter $\widehat{\rho}_{w,T}$ in a similar way as $\rho_{w,T}$ is defined and show that $2\widehat{\rho}_{w,T} + k + 1$ is an 8-approximation of $\delta(G)$ with an additive error of $3k + 1$.

▶ **Proposition 11.** *Given a graph G, a BFS-tree T rooted at a vertex w, and a matrix $\widehat{D}$ such that $d(x, y) \leq \widehat{D}(x, y) \leq d(x, y) + k$, we can compute in time $O(n^2)$ a value $\widehat{\rho}_{w,T}$ such that $\delta(G) \leq 2\widehat{\rho}_{w,T} + k + 1 \leq 8\delta(G) + 3k + 1$.*

Interestingly, $\rho_{w,T}$ can also be defined in terms of a distance approximation parameter. Consider a geodesic space X and a GS-tree T rooted at some point w, and let $\rho = \rho_{w,T}$. For a point $x \in X$ and $r \in \mathbb{R}^+$, denote by $x(r)$ the unique point of $[w, x]_T$ at distance r from w if $r < d(w, x)$ and the point x if $r \geq d(w, x)$. For any x, y and $\epsilon \in \mathbb{R}^+$, let $r_{xy}(\epsilon) := \sup\{r : d(x(r'), y(r')) \leq \epsilon \text{ for any } 0 \leq r' \leq r\}$. This supremum is a maximum because the function $r' \mapsto d(x(r'), y(r'))$ is continuous. Observe that by Proposition 6, $\rho = \inf\{\epsilon : r_{xy}(\epsilon) \geq (x|y)_w \text{ for all } x, y\}$.

Denote by $x_y(\epsilon)$ (respectively, $y_x(\epsilon)$) the point of $[x, w]_T$ (respectively, of $[w, y]_T$) at distance $r_{xy}(\epsilon)$ from w. Let $\widehat{d}_\epsilon(x, y) = d(x, x_y(\epsilon)) + \epsilon + d(y_x(\epsilon), y)$. By the triangle inequality, $d(x, y) \leq d(x, x_y(\epsilon)) + d(x_y(\epsilon), y_x(\epsilon)) + d(y_x(\epsilon), y) \leq \widehat{d}_\epsilon(x, y)$. Observe that for any ϵ and for any x, y, we have $r_{xy}(\epsilon) \geq (x|y)_w$ if and only if $d(x, x_y(\epsilon)) + d(y_x(\epsilon), y) \leq d(x, y)$, i.e., if and only if $d(x, y) \leq \widehat{d}_\epsilon(x, y) \leq d(x, y) + \epsilon$. Consequently, $\rho = \inf\{\epsilon : d(x, y) \leq \widehat{d}_\epsilon(x, y) \leq d(x, y) + \epsilon \text{ for all } x, y\}$.

When we consider a graph G with a BFS-tree T rooted at some vertex w, we have similar results. For a vertex x, we define $x(r)$ as before when r is an integer and for vertices x, y, we define $r_{xy}(\epsilon) := \max\{r \in \mathbb{N} : d(x(r'), y(r')) \leq \epsilon \text{ for any } 0 \leq r' \leq r\}$. Since $\rho = \inf\{\epsilon : r_{xy}(\epsilon) \geq \lfloor (x|y)_w \rfloor \text{ for all } x, y\}$, we get that $d(x, y) \leq \widehat{d}_\rho(x, y) + 1 \leq d(x, y) + \rho + 1$.

▶ **Proposition 12.** *If the distance matrix D of a graph G is unknown but the kth power graph G^k of G is given for $k \geq \rho_{w,T}$, then one can approximate the distance matrix D of G in optimal $O(n^2)$ time with an additive term depending only on k.*

Proof. With G^k at hand, for a fixed vertex $x \in X$ the values of $r_{xy}(k)$ and $\widehat{d}_k(x,y)$, for every $y \in X$, can be computed in linear time using a simple traversal of the BFS-tree T. ◀

4.2 Fast approximation of thinness, insize, and slimness

Using Proposition 3 and Theorem 7, we also get the following corollary.

▶ **Corollary 13.** *For a graph G and a BFS-tree T rooted at a vertex w, $\tau(G) \leq 8\rho_{w,T} + 4 \leq 8\tau(G) + 4$ and $\varsigma(G) \leq 6\rho_{w,T} + 3 \leq 24\varsigma(G) + 3$. Consequently, an 8-approximation (with additive surplus 4) of the thinness $\tau(G)$ and a 24-approximation (with additive surplus 3) of the slimness $\varsigma(G)$ can be found in $O(n^2)$ time.*

Consider a collection $\mathcal{T} = (T_w)_{w \in V}$ of trees where for each w, T_w is an arbitrary BFS-tree rooted at w, and let $\rho_{\mathcal{T}} = \max_{w \in V} \rho_{w,T_w}$. Since for each w, ρ_{w,T_w} can be computed in $O(n^2)$ time, $\rho_{\mathcal{T}}$ can be computed in $O(n^3)$ time. We stress that for any fixed $w \in V$, $\delta_w(G)$ can be also computed in $O(n^3)$ time. Furthermore, by Proposition 1, $\delta_w(G)$ gives a 2-approximation of the hyperbolicity $\delta(G)$ of G. In what follows, we present similar complexity approximations for $\varsigma(G)$ and $\tau(G)$.

To get a better bound for $\varsigma(G)$, we need to involve one more parameter. Let u and v be arbitrary vertices of G and $T_u \in \mathcal{T}$ be the BFS-tree rooted at u. Let also $(u = u_0, u_1, \ldots, u_\ell = v)$ be the path of T_u joining u with v. Define $\kappa_{T_u}(u,v) := \max\{d(a,u_i) : a \in I(u,v), d(a,u) = i\}$ and $\kappa_{\mathcal{T}} := \max\{\kappa_{T_u}(u,v) : u,v \in V\}$. Note that $\kappa_{\mathcal{T}} \leq \kappa(G)$ and that $\kappa_{\mathcal{T}}$ can be computed in $O(n^3)$ time and $O(n^2)$ space. Observe also that for any u,v, $\kappa_{T_u}(u,v) \leq \rho_{u,T_u}$ and thus $\kappa_{\mathcal{T}} \leq \rho_{\mathcal{T}}$.

▶ **Proposition 14.** *For a graph G and a collection of BFS-trees $\mathcal{T} = (T_w)_{w \in V}$, $\iota(G) = \tau(G) \leq \rho_{\mathcal{T}} + 2\kappa_{\mathcal{T}} \leq 3\rho_{\mathcal{T}} \leq 3\tau(G)$ and $\varsigma(G) \leq \rho_{\mathcal{T}} + 2\kappa_{\mathcal{T}} \leq 8\varsigma(G)$. Consequently, a 3-approximation of the thinness $\tau(G)$ and an 8-approximation of the slimness $\varsigma(G)$ can be found in $O(n^3)$ time and $O(n^2)$ space.*

Proof. Pick any geodesic triangle $\Delta(x,y,w)$ with sides $[x,y]$, $[x,w]$ and $[y,w]$. Let $[x,w]_T$ and $[y,w]_T$ be the corresponding geodesics of the BFS-tree T for vertex w. Consider the vertices $x_y \in [x,w]_T, y_x \in [w,y]_T$ and vertices $a \in [x,w], b \in [y,w]$ with $d(w,x_y) = d(w,y_x) = d(w,a) = d(w,b) = \lfloor (x|y)_w \rfloor$. We know that $d(x_y,y_x) \leq \rho_{\mathcal{T}}$. Since $(x|a)_w = d(a,w)$ and $(y|b)_w = d(b,w)$, $d(a,x_y) \leq \kappa_{T_w}(w,x) \leq \kappa_{\mathcal{T}}$ and $d(b,y_x) \leq \kappa_{T_w}(w,y) \leq \kappa_{\mathcal{T}}$. Hence, $d(a,b) \leq \rho_{\mathcal{T}} + 2\kappa_{\mathcal{T}}$. Repeating this argument for vertices x and y and their BFS-trees, we get that the insize of $\Delta(x,y,w)$ is at most $\rho_{\mathcal{T}} + 2\kappa_{\mathcal{T}}$. So $\tau(G) \leq \rho_{\mathcal{T}} + 2\kappa_{\mathcal{T}} \leq 3\rho_{\mathcal{T}} \leq \tau(G)$ and by Proposition 3, $\varsigma(G) \leq \tau(G) \leq \rho_{\mathcal{T}} + 2\kappa_{\mathcal{T}} \leq \tau(G) + 2\kappa(G) \leq 8\varsigma(G)$. ◀

5 Exact computation

In this section, we provide exact algorithms for computing the slimness $\varsigma(G)$, the thinness $\tau(G)$, and the insize $\iota(G)$ of a given graph G. The algorithm computing $\tau(G) = \iota(G)$ runs in $O(n^2m)$ time and the algorithm computing $\varsigma(G)$ runs in $\widehat{O}(n^2m + n^4/\log^3 n)$ time; both algorithms use $O(n^2)$ space. When the graph is dense (i.e., $m = \Omega(n^2)$), that stays of the same order of magnitude as the best-known algorithms for computing $\delta(G)$ in practice (see [6]), but when the graph is not so dense (i.e., $m = o(n^2)$), our algorithms run in $o(n^4)$ time. In contrast to this result, the existing algorithms for computing $\delta(G)$ exactly are not sensitive to the density of the input. We also show that the minsize $\rho_-(G)$ of a given graph G cannot be approximated with a factor strictly better than 2 unless P = NP. The results of this section are summarized by the following theorem:

▶ **Theorem 15.** *For a graph $G = (V, E)$ with n vertices and m edges, the following holds:*
(1) *the thinness $\tau(G)$ and the insize $\iota(G)$ of G can be computed in $O(n^2m)$ time;*
(2) *the slimness $\varsigma(G)$ of G can be computed in $\widehat{O}(n^2m + n^4/\log^3 n)$ time ;*
(3) *Deciding whether the minsize $\rho_-(G)$ of G is at most 1 is NP-complete.*

5.1 Exact computation of thinness and insize

We present an algorithm to compute $\tau(G) = \iota(G)$ that runs in time $O(n^2m)$. We first compute the distance matrix D of G in time $O(mn)$. To compute $\tau(G)$, we introduce the "pointed thinness" $\tau_w(G)$ of a given vertex w. For a fixed vertex w, let $\tau_w(G) = \max\big\{d(x', y') : \exists x, y \in V$ such that $x' \in I(w, x), y' \in I(w, y)$, and $d(w, x') = d(w, y') \leq (x|y)_w\big\}$. Observe that for any BFS-tree T rooted at w, we have $\rho_{w,T} \leq \tau_w(G) \leq \tau(G)$, and thus by Corollary 13, $\tau_w(G)$ is an 8-approximation (with additive surplus 4) of $\tau(G)$. Since $\tau(G) = \max_{w\in V} \tau_w(G)$, in order to prove Theorem 15(1), it is sufficient to describe an algorithm computing $\tau_w(G)$ in $O(mn)$. Let $\tau_{w,x}(G) = \max\big\{d(x', y') : x' \in I(w, x)$ and $\exists y \in V$ such that $y' \in I(w, y)$ and $d(w, x') = d(w, y') \leq (x|y)_w\big\}$ and observe that $\tau_w(G) = \max_{x\in V} \tau_{w,x}(G)$.

For every ordered pair w, x and every vertex z, let $g_z(w, x) = \max\big\{d(x', z) : x' \in I(w, x)$ and $d(w, x') = d(w, z)\big\}$. Observe that $\tau_{w,x} = \max\big\{g_z(w, x) : \exists y$ such that $z \in I(w, y)$ and $d(w, z) \leq (x|y)_w\big\}$.

▶ **Lemma 16.** *For any fixed $w, z \in V$, one can compute the values of $g_z(w, x)$ for all $x \in V$ in $O(m)$ time.*

Proof. In order to compute $g_z(w, x)$, we use the following recursive formula: $g_z(w, x) = 0$ if $d(w, x) < d(w, z)$, $g_z(w, x) = d(x, z)$ if $d(w, x) = d(w, z)$, and $g_z(w, x) = \max\big\{g_z(w, x') : x' \in N(x)$ and $d(w, x') = d(w, x) - 1\big\}$ otherwise. Given the distance matrix D, for any $x \in V$, we can compute $\{x' \in N(x) : d(w, x') = d(w, x) - 1\}$ in $O(\deg(x))$ time. Therefore, using a standard dynamic programming approach, we can compute the values $g_z(w, x)$ for all $x \in V$ in $O(\sum_x \deg(x)) = O(m)$ time. ◀

Let $h_{w,x}(z) = \max\big\{(x|y)_w : z \in I(w, y)\big\}$ and observe that $\tau_{w,x}(G) = \max\big\{g_z(w, x) : d(w, z) \leq h_{w,x}(z)\big\}$. Note that if $w, x \in V$ are fixed, then $h_{w,x}(z)$ satisfies the following recursive formula: $h_{w,x}(z) = \max\big\{(x|z)_w, h'_{w,x}(z)\big\}$ where $h'_{w,x}(z) = \max\big\{h_{w,x}(z') : z' \in N(z)$ and $d(w, z') = d(w, z) + 1\big\}$. If we order the vertices of V by non-increasing distance to w, using dynamic programming, we can compute the values of $h_{w,x}(z)$ for all z in $O(\sum_z \deg(z)) = O(m)$ time.

We can thus compute the values $g_z(w, x)$ and $h_{w,x}(z)$ for all $x, z \in V$ in $O(mn)$ time. Then for every fixed w, x, we can compute $\tau_{w,x}(G) = \max\big\{g_z(w, x) : d(w, z) \leq h_{w,x}(z)\}$ in $O(n)$ time, and consequently we can compute $\tau_w(G) = \max_x \tau_{w,x}(G)$ in $O(mn)$ time.

5.2 Exact computation of slimness

To prove Theorem 15(2), we introduce the "pointed slimness" $\varsigma_w(G)$ of a given vertex w. Formally, $\varsigma_w(G)$ is the least integer k such that, in any geodesic triangle $\Delta(x, y, z)$ such that $w \in [x, y]$, we have $d(w, [x, z] \cup [y, z]) \leq k$. Note that $\varsigma_w(G)$ cannot be used to approximate $\varsigma(G)$ (that is in sharp contrast with $\delta_w(G)$ or $\tau_w(G)$). In particular, $\varsigma_w(G) = 0$ whenever w is a pending vertex of G (or, more generally, a simplicial vertex of G). On the other hand, we have $\varsigma(G) = \max_{w\in V} \varsigma_w(G)$. Therefore, in order to prove Theorem 15(2), it is sufficient to describe an algorithm computing $\varsigma_w(G)$ that runs in $\widehat{O}(nm + n^3/\log^3 n)$ time for every

w. For every $x, z \in V$ we set $p_w(x,z)$ to be the least integer k such that, for every geodesic $[x,z]$, we have $d(w,[x,z]) \leq k$. Observe that $\varsigma_w(G) \leq k$ if and only if for all $x, y \in V$ such that $w \in I(x,y)$, and any $z \in V$, $\min\{p_w(x,z), p_w(y,z)\} \leq k$.

As before, we assume that the distance matrix D of G has been already computed. The algorithm for computing $\varsigma_w(G)$ proceeds in two phases. We first compute $p_w(x,z)$ for every $x, z \in V$. Second, we seek for a triple (x,y,z) such that $w \in I(x,y)$ and $\min\{p_w(x,z), p_w(y,z)\}$ is maximized.

For any fixed $w, x \in V$, observe that $p_w(x,z)$ satisfies the following recursive formula: $p_w(x,x) = d(w,x)$ and for any $z \neq x$, $p_w(x,z) = \min\{d(w,z), p'_w(x,z)\}$ where $p'_w(x,z) = \max\{p_w(x,z') : z' \in N(z) \text{ and } d(x,z') = d(x,z) - 1\}$. Using dynamic programming, one can compute the values $p_w(x,z)$ for all $z \in V$ in $O(m)$ time. Consequently, we can compute the values $p_w(y,z)$ for all $y, z \in V$ in $O(mn)$ time, and then, we can compute $\varsigma_w(G)$ in $O(n^3)$ time by enumerating all possible triples $x, y, z \in V$ such that $w \in I(x,y)$ and keeping one maximizing $\min\{p_w(x,z), p_w(y,z)\}$.

We can improve the running time by reducing the problem to Triangle Detection as follows. Given a fixed integer k, the graph $\Gamma^w_\varsigma[k]$ has vertex set $V_1 \cup V_2 \cup V_3$, with every set V_i being a copy of $V \setminus \{w\}$. There is an edge between $x_1 \in V_1$ and $y_2 \in V_2$ if and only if the corresponding vertices $x, y \in V$ satisfy $w \in I(x,y)$. Furthermore, there is an edge between $x_1 \in V_1$ and $z_3 \in V_3$ (respectively, between $y_2 \in V_2$ and $z_3 \in V_3$) if and only if we have $p_w(x,z) > k$ (respectively, $p_w(y,z) > k$). It is easy to see that $\varsigma_w(G) \leq k$ if and only if $\Gamma^w_\varsigma[k]$ is triangle-free. Once the distance matrix of G and the values $p_w(y,z)$ for all $y, z \in V$ have been computed, we can construct $\Gamma^w_\varsigma[k]$ in $O(n^2)$ time. Since Triangle Detection can be solved in $\widehat{O}(n^3/\log^4 n)$ time [35], we can decide whether $\varsigma_w(G) \leq k$ in the same time, and by performing binary search, we can compute $\varsigma_w(G)$ in $\widehat{O}(n^3/\log^3 n)$ time.

5.3 Approximating the minsize is hard

We now prove Theorem 15(3). Note that if we are given a BFS-tree T rooted at a vertex w, we can easily check whether $\rho_{w,T} \leq 1$, and thus deciding whether $\rho_-(G) \leq 1$ is in NP. In order to prove that this problem is NP-hard, we do a reduction from Sat. Let Φ be a Sat formula with m clauses $c_1, c_2, \ldots, c_m$ and n variables $x_1, x_2, \ldots, x_n$. Let $X = \{x_1, \overline{x}_1, \ldots, x_n, \overline{x}_n\}$. For simplicity, in what follows, we denote $x_i, \overline{x}_i$ by ℓ_{2i-1}, ℓ_{2i}. Let $C = \{c_1, \ldots, c_m\}$ be the clause-set of Φ. Finally, let w and $V = \{v_1, v_2, \ldots, v_{2n}\}$ be additional vertices. We construct a graph G_Φ with $V(G_\Phi) = \{w\} \cup V \cup X \cup C$ and where $E(G_\Phi)$ is defined as follows:

- $N(w) = V$ and V is a clique,
- for every i, i', v_i and $l_{i'}$ are adjacent if and only if $i = i'$;
- for every i, i', ℓ_i and $\ell_{i'}$ are adjacent if and only if $\ell_{i'} \neq \overline{\ell}_i$;
- for every i, j, v_i and c_j are not adjacent;
- for every i, j, ℓ_i and c_j are adjacent if and only if $\ell_i \in c_j$;
- for every j, j', $c_j, c_{j'}$ are adjacent if and only if $c_j, c_{j'}$ intersect in exactly one literal.

We can show that we can preprocess the formula Φ in polynomial time such that:

(1) for every BFS-tree T rooted at $u \neq w$, $\rho_{u,T} \geq 2$,
(2) for any BFS-tree T rooted at w, for any $t, u \in V$, if $d(t_u, u_t) \geq 2$, then $t, u \in C$,
(3) for every $c, c' \in C$, $d(c,c') \leq 2$.

Now, for every $c, c' \in C$, observe that $\lfloor (c|c')_w \rfloor = 2$ and thus $c_{c'}, c'_c \in X$. Consequently, $\rho_{w,T} \leq 1$ if and only if $d(c_{c'}, c'_c) \leq 1$ for all $c, c' \in C$, i.e., if and only if $c_{c'} \neq \overline{c'_c}$ for all $c, c' \in C$. Therefore, there exists a tree T rooted at w such that $\rho_{w,T} = 1$ if and only if there exists a satisfying assignment for Φ.

References

1 M. Abu-Ata and F.F. Dragan. Metric tree-like structures in real-world networks: an empirical study. *Networks*, 67(1):49–68, 2016.

2 A.B. Adcock, B.D. Sullivan, and M.W. Mahoney. Tree-like structure in large social and information networks. In *ICDM*, pages 1–10. IEEE Computer Society, 2013.

3 J.M. Alonso, T. Brady, D. Cooper, V. Ferlini, M. Lustig, M. Mihalik, M. Shapiro, and H. Short. Notes on word hyperbolic groups. In E. Ghys, A. Haefliger, and A. Verjovsky, editors, *Group Theory from a Geometrical Viewpoint*, ICTP Trieste 1990, pages 3–63. World Scientific, 1991.

4 A.M. Ben-Amram. The Euler path to static level-ancestors. *CoRR*, abs/0909.1030, 2009.

5 M.A. Bender and M Farach-Colton. The level ancestor problem simplified. *Theor. Comput. Sci.*, 321(1):5–12, 2004.

6 M. Borassi, D. Coudert, P. Crescenzi, and A. Marino. On computing the hyperbolicity of real-world graphs. In *ESA*, volume 9294 of *Lecture Notes in Computer Science*, pages 215–226. Springer, 2015.

7 M. Borassi, P. Crescenzi, and M. Habib. Into the square: On the complexity of some quadratic-time solvable problems. *Electr. Notes Theor. Comput. Sci.*, 322:51–67, 2016.

8 B.H. Bowditch. Notes on Gromov's hyperbolicity criterion for path-metric spaces. In E. Ghys, A. Haefliger, and A. Verjovsky, editors, *Group Theory from a Geometrical Viewpoint*, ICTP Trieste 1990, pages 64–167. World Scientific, 1991.

9 M.R. Bridson and A. Haefliger. *Metric Spaces of Non-Positive Curvature*, volume 319 of *Grundlehren der mathematischen Wissenschaften*. Springer-Verlag, Berlin, 1999.

10 J. Chalopin, V. Chepoi, F.F. Dragan, G. Ducoffe, A. Mohammed, and Y. Vaxès. Fast approximation and exact computation of negative curvature parameters of graphs. *CoRR*, abs/1803.06324, 2018.

11 J. Chalopin, V. Chepoi, P. Papasoglu, and T. Pecatte. Cop and robber game and hyperbolicity. *SIAM J. Discrete Math.*, 28(4):1987–2007, 2014.

12 V. Chepoi, F.F. Dragan, B. Estellon, M. Habib, and Y. Vaxès. Diameters, centers, and approximating trees of delta-hyperbolicgeodesic spaces and graphs. In *Symposium on Computational Geometry*, pages 59–68. ACM, 2008.

13 V. Chepoi, F.F. Dragan, B. Estellon, M. Habib, Y. Vaxès, and Yang Xiang. Additive spanners and distance and routing labeling schemes for hyperbolic graphs. *Algorithmica*, 62(3-4):713–732, 2012.

14 V. Chepoi, F.F. Dragan, and Y. Vaxès. Core congestion is inherent in hyperbolic networks. In *SODA*, pages 2264–2279. SIAM, 2017.

15 V. Chepoi and B. Estellon. Packing and covering δ-hyperbolic spaces by balls. In *APPROX-RANDOM*, volume 4627 of *Lecture Notes in Computer Science*, pages 59–73. Springer, 2007.

16 N. Cohen, D. Coudert, and A. Lancin. On computing the Gromov hyperbolicity. *ACM Journal of Experimental Algorithmics*, 20:1.6:1–1.6:18, 2015.

17 D. Coudert and G. Ducoffe. Recognition of C_4-free and 1/2-hyperbolic graphs. *SIAM J. Discrete Math.*, 28(3):1601–1617, 2014.

18 D. Coudert, G. Ducoffe, and A. Popa. Fully polynomial FPT algorithms for some classes of bounded clique-width graphs. In *SODA*, pages 2765–2784. SIAM, 2018.

19 B. DasGupta, M. Karpinski, N. Mobasheri, and F. Yahyanejad. Effect of Gromov-hyperbolicity parameter on cuts and expansions in graphs and some algorithmic implications. *Algorithmica*, 80(2):772–800, 2018.

20 T. Delzant and M. Gromov. Courbure mésoscopique et théorie de la toute petite simplification. *J. Topol.*, 1:804–836, 2008.

21 R. Duan. Approximation algorithms for the Gromov hyperbolicity of discrete metric spaces. In *LATIN*, volume 8392 of *Lecture Notes in Computer Science*, pages 285–293. Springer, 2014.

22 K. Edwards, W.S. Kennedy, and I. Saniee. Fast approximation algorithms for p-centers in large δ-hyperbolic graphs. In *WAW*, volume 10088 of *Lecture Notes in Computer Science*, pages 60–73, 2016.

23 T. Fluschnik, C. Komusiewicz, G.B. Mertzios, A. Nichterlein, R. Niedermeier, and N. Talmon. When can graph hyperbolicity be computed in linear time? In *WADS*, volume 10389 of *Lecture Notes in Computer Science*, pages 397–408. Springer, 2017.

24 H. Fournier, A. Ismail, and A. Vigneron. Computing the Gromov hyperbolicity of a discrete metric space. *Inf. Process. Lett.*, 115(6-8):576–579, 2015.

25 E. Ghys and P. de la Harpe (eds). *Les groupes hyperboliques d'après M. Gromov*, volume 83 of *Progress in Mathematics*. Birkhäuser, 1990.

26 M. Gromov. Hyperbolic groups. In S. Gersten, editor, *Essays in Group Theory*, volume 8 of *Math. Sci. Res. Inst. Publ.*, pages 75–263. Springer, New York, 1987.

27 M.F. Hagen. Weak hyperbolicity of cube complexes and quasi-arboreal groups. *J. Topology*, 7(2):385–418, 2014.

28 W.S. Kennedy, I. Saniee, and O. Narayan. On the hyperbolicity of large-scale networks and its estimation. In *BigData*, pages 3344–3351. IEEE, 2016.

29 O. Narayan and I. Saniee. Large-scale curvature of networks. *Phys. Rev. E*, 84:066108, 2011.

30 P. Papasoglu. An algorithm detecting hyperbolicity. In *Geometric and computational perspectives on infinite groups (Minneapolis, MN and New Brunswick, NJ, 1994)*, volume 25 of *DIMACS - Series in Discrete Mathematics and Theoretical Computer Science*, pages 193–200. 1996.

31 N. Polat. On infinite bridged graphs and strongly dismantlable graphs. *Discrete Mathematics*, 211:153–166, 2000.

32 Y. Shavitt and T. Tankel. Hyperbolic embedding of internet graph for distance estimation and overlay construction. *IEEE/ACM Trans. Netw.*, 16(1):25–36, 2008.

33 M. Soto. *Quelques propriétés topologiques des graphes et applications à Internet et aux réseaux*. PhD thesis, Université Paris Diderot, 2011.

34 K. Verbeek and S. Suri. Metric embedding, hyperbolic space, and social networks. In *Symposium on Computational Geometry*, pages 501–510. ACM, 2014.

35 H. Yu. An improved combinatorial algorithm for boolean matrix multiplication. In *ICALP(1)*, volume 9134 of *Lecture Notes in Computer Science*, pages 1094–1105. Springer, 2015.

Tree Drawings Revisited

Timothy M. Chan
Department of Computer Science, University of Illinois at Urbana-Champaign, USA
tmc@illinois.edu

Abstract

We make progress on a number of open problems concerning the area requirement for drawing trees on a grid. We prove that

1. every tree of size n (with arbitrarily large degree) has a straight-line drawing with area $n2^{O(\sqrt{\log\log n \log\log\log n})}$, improving the longstanding $O(n \log n)$ bound;
2. every tree of size n (with arbitrarily large degree) has a straight-line upward drawing with area $n\sqrt{\log n}(\log\log n)^{O(1)}$, improving the longstanding $O(n \log n)$ bound;
3. every binary tree of size n has a straight-line orthogonal drawing with area $n2^{O(\log^* n)}$, improving the previous $O(n \log\log n)$ bound by Shin, Kim, and Chwa (1996) and Chan, Goodrich, Kosaraju, and Tamassia (1996);
4. every binary tree of size n has a straight-line order-preserving drawing with area $n2^{O(\log^* n)}$, improving the previous $O(n \log\log n)$ bound by Garg and Rusu (2003);
5. every binary tree of size n has a straight-line orthogonal order-preserving drawing with area $n2^{O(\sqrt{\log n})}$, improving the $O(n^{3/2})$ previous bound by Frati (2007).

2012 ACM Subject Classification Theory of computation → Computational geometry, Mathematics of computing → Trees, Human-centered computing → Graph drawings

Keywords and phrases graph drawing, trees, recursion

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.23

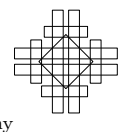
Related Version A full version of this paper is available at `https://arxiv.org/abs/1803.07185`.

1 Introduction

Drawing graphs with small area has been a subject of intense study in combinatorial and computational geometry for more than two decades [11, 12]. The goal is to determine worst-case bounds on the area needed to draw any n-vertex graph in a given class, subject to certain drawing criteria, where vertices are mapped to points on an integer grid $\{1, \ldots, W\} \times \{1, \ldots, H\}$, and the *area* of the drawing is defined to be the width W times the height H. All drawings in this paper are required to be *planar*, where edge crossings are not allowed. All our results will be about *straight-line* drawings, where edges are drawn as straight line segments, although *poly-line* drawings that allow bends along the edges have also received considerable attention.

It is well known [10, 23] that every planar graph of size n has a straight-line drawing with area $O(n^2)$ (with width and height $O(n)$), and this bound is asymptotically tight in the worst case. Much research is devoted to understanding which subclasses of planar graphs admit subquadratic-area drawings, and obtaining tight area bounds for such classes.

Drawing arbitrary trees. Among the simplest is the class of all trees. As hierarchical structures occur naturally in many areas (from VLSI design to phylogeny), visualization of trees is of particular interest. Although there have been numerous papers on tree drawings



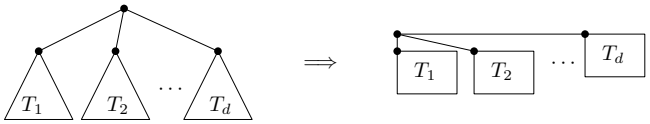


Figure 1 The "standard" algorithm to produce a straight-line upward drawing of any tree of size n, with width at most n and height at most $\lceil \log n \rceil$: reorder the subtrees so that T_d is the largest, then recursively draw $T_1, \ldots, T_d$.

(e.g., [2, 4, 5, 6, 7, 8, 9, 13, 14, 15, 16, 17, 18, 19, 20, 22, 25, 26, 24, 27, 28]), the most basic question of determining the worst-case area needed to draw arbitrary trees, without any additional criteria other than being planar and straight-line, is surprisingly still open.

An $O(n \log n)$ area upper bound is folklore and can be obtained by a straightforward recursive algorithm, as described in Figure 1, which we will refer to as the *standard* algorithm (the earliest reference was perhaps Shiloach's 1976 thesis [24, page 94]; see also Crescenzi, Di Battista, and Piperno [8] for the same algorithm for binary trees). The algorithm gives linear width and logarithmic height. An analogous algorithm, with x and y coordinates swapped, gives logarithmic width and linear height.

However, no single improvement to the $O(n \log n)$ bound has been found for general trees. No improvement is known even if drawings are relaxed to be poly-line!

In an early SoCG'93 paper by Garg, Goodrich, and Tamassia [15], it was shown that linear area is attainable for poly-line drawings of trees with degree bounded by $O(n^{1-\varepsilon})$ for any constant $\varepsilon > 0$. Later, Garg and Rusu [18, 17] obtained a similar result for straight-line drawings for degree up to $O(n^{1/2-\varepsilon})$.[1] These approaches do not give good bounds when the maximum degree is linear.

To understand why unbounded degree can pose extra challenges, consider the extreme case when the tree is a star of size n, and we want to draw it on an $O(\sqrt{n}) \times O(\sqrt{n})$ grid. A solution is not difficult if we use the fact that relatively prime pairs are abundant, but most tree drawing algorithms use geometric divide-and-conquer strategies that do not seem compatible with such number-theoretic ideas.

New results. Our first main result is the first $o(n \log n)$ area upper bound for straight-line drawings of arbitrary trees: the bound is $n2^{O(\sqrt{\log \log n \log \log \log n})}$, which in particular is better than $O(n \log^{\varepsilon} n)$ for any constant $\varepsilon > 0$.

Even to those who care less about refining logarithmic factors, our method has one notable advantage: it can give drawings achieving a full range of width–height tradeoffs (in other words, a full range of aspect ratios). For example, we can simultaneously obtain width and height $\sqrt{n}2^{O(\sqrt{\log n \log \log n})}$. Although the extra factor is now superpolylogarithmic, the result is still new. In contrast, the standard algorithm (Figure 1) produces only narrow drawings, whereas the previous approaches of Garg et al. [15, 18] provided width–height tradeoffs but inherently cannot give near $\sqrt{n}$ perimeter if degree exceeds $\sqrt{n}$.

For rooted trees, it is natural to consider *upward* drawings, where the y-coordinate of each node is greater than or equal to the y-coordinate of each child. The drawing obtained by the standard algorithm is upward. We obtain the first $o(n \log n)$ area bound for straight-line upward drawings of arbitrary trees as well: the bound is near $O(n\sqrt{\log n})$, ignoring small

[1] It is not clear to this author if their analysis assumed a much stronger property, that every subtree of size m has degree at most $O(m^{1/2-\varepsilon})$.

Table 1 Worst-case area bounds for straight-line drawings of *arbitrary trees*. (In all tables, c denotes some constant, and Θ denotes tight results that have matching lower bounds.)

	non-order-preserving	order-preserving
non-upward	$O(n \log n)$ by standard alg'm $O(nc^{\sqrt{\log\log n \log\log\log n}})$ **new**	$O(n \log n)$ by Garg–Rusu'03 [16]
upward	$O(n \log n)$ by standard alg'm $O(n\sqrt{\log n}\log^c \log n)$ **new**	$O(nc^{\sqrt{\log n}})$ by Chan'99 [6]
strictly upward	$\Theta(n \log n)$ by standard alg'm [8]	$O(nc^{\sqrt{\log n}})$ by Chan'99 [6]

Table 2 Worst-case area bounds for straight-line drawings of *binary trees*.

	non-order-preserving	order-preserving
non-upward	$\Theta(n)$ by Garg–Rusu'04 [18]	$O(n \log\log n)$ by Garg–Rusu'03 [16] $O(nc^{\log^* n})$ **new**
upward	$O(n \log\log n)$ by Shin–Kim–Chwa'96 [25]	$O(n^{1.48})$ by Chan'99 [6] $O(nc^{\sqrt{\log n}})$ by Chan'99 [6] $O(n \log n)$ by Garg–Rusu'03 [16]
strictly upward	$\Theta(n \log n)$ by standard alg'm [8]	$O(n^{1.48})$ by Chan'99 [6] $O(nc^{\sqrt{\log n}})$ by Chan'99 [6] $\Theta(n \log n)$ by Garg–Rusu'03 [16]

log log factors. (See Table 1.)

These results represent significant progress towards Open Problems 5, 6, 17, and 18 listed in Di Battista and Frati's recent survey [12].

We will describe the near-$O(n\sqrt{\log n})$ upward algorithm first, in Section 2, which prepares us for the more involved $n2^{O(\sqrt{\log\log n \log\log\log n})}$ non-upward algorithm in Section 3.

Drawing binary trees. Next we turn to drawings of binary trees, where there has been a large body of existing work, due to the many combinations of aesthetic criteria that may be imposed. We may consider

- *upward* drawings, as defined earlier;
- *strictly upward* drawings, where the y-coordinate of each node is strictly greater the y-coordinate of each child;
- *order-preserving* drawings, where the order of children of each node v is preserved, i.e., the parent, the left child, and the right child of v appear in counterclockwise order around v;
- *orthogonal* drawings, where all edges are drawn with horizontal or vertical line segments.

Tables 2–3 summarize the dizzying array of known results on straight-line drawings. (To keep the table size down, we omit numerous other results on poly-line drawings, and on special subclasses of balanced trees. See Di Battista and Frati's survey [12] for more.)

Table 3 Worst-case area bounds for straight-line *orthogonal* drawings of *binary trees*. (Strictly upward drawings are not possible here.)

	non-order-preserving		order-preserving	
non-upward	$O(n \log\log n)$	by Chan–Goodrich–Kosaraju–Tamassia'96 [7] & Shin–Kim–Chwa'96 [25]	$O(n^{3/2})$	Frati'07 [13]
	$O(nc^{\log^* n})$	**new**	$O(nc^{\sqrt{\log n}})$	**new**
upward	$\Theta(n \log n)$	by standard alg'm [8]	$\Theta(n^2)$	

New results. In this paper, we concentrate on two of the previous $O(n \log\log n)$ entries in the table. In 1996, Shin, Kim, and Chwa [25] and Chan et al. [7] independently obtained $O(n \log\log n)$-area algorithms for straight-line orthogonal drawings of binary trees; a few years later, Garg and Rusu [16] adapted their technique to obtain similar results for straight-line (non-orthogonal) order-preserving drawings. We improve the area bound for both types of drawings to *almost* linear: $n2^{O(\log^* n)}$, where $\log^*$ denotes the iterated logarithm. (We can also obtain width–height tradeoffs for these drawings.)

Although improving $\log\log n$ to iterated logarithm may not come as a total surprise, the problem for straight-line orthogonal drawings has resisted attack for 20 years. (Besides, improvement should not be taken for granted, since there is at least one class of drawings for which $\Theta(n \log\log n)$ turns out to be tight: poly-line upward orthogonal drawings of binary trees [15].)

We have additionally one more result on straight-line orthogonal order-preserving drawings of binary trees: in 2007, Frati [13] presented an $O(n^{3/2})$-area algorithm. We improve the bound to $n2^{O(\sqrt{\log n})}$, which in particular is better than $O(n^{1+\varepsilon})$ for any constant $\varepsilon > 0$.

These results represent significant progress towards Open Problems 9, 12, and 14 listed in Di Battista and Frati's survey [12].

(The author has obtained still more new results, on a special class of so-called *LR drawings* of binary trees [6, 14], making progress on Open Problem 10 in the survey, which will be reported later elsewhere.)

We will describe the $n2^{O(\log^* n)}$ algorithm for orthogonal drawings first, in Section 4; the algorithm for non-orthogonal order-preserving drawings is similar, and is described in the full paper. The $n2^{O(\sqrt{\log n})}$ algorithm for orthogonal order-preserving drawings is different and is also deferred to the full paper.

Techniques. Various tree-drawing techniques have been identified in the large body of previous work, and we will certainly draw upon some of these existing techniques in our new algorithms—in particular, the use of "skewed" centroids for divide-and-conquer in trees (see Section 2 for the definition), and height–width tradeoffs to obtain better area bounds.

However, as the unusual bounds would suggest, our $n2^{O(\sqrt{\log\log n \log\log\log n})}$ and our $n2^{O(\log^* n)}$ algorithms will require new forms of recursion and bootstrapping.

Our $n2^{O(\sqrt{\log\log n \log\log\log n})}$ result for arbitrary trees requires novelty not just in fancier recurrences, but also in geometric insights. All existing divide-and-conquer algorithms for tree drawings divide a given tree into subtrees and recursively draw different subtrees in different, disjoint axis-aligned bounding boxes. We will depart from tradition and draw some parts of the tree in distorted grids inside narrow sectors, which are remapped to regular grids through

affine transformations every time we bootstrap. The key is a geometric observation that any two-dimensional convex set (however narrow) containing a large number of integer points must contain a large subset of integer points forming a grid after affine transformation (with unspecified aspect ratio). The proof of the observation follows from well known facts about lattices and basis reduction (by Gauss)—a touch of elementary number theory suffices. We are not aware of previous applications of this geometric observation, which seems potentially useful for graph drawing on grids in general.

Our $n2^{O(\log^* n)}$ result is noteworthy, because occurrences of iterated logarithm are rare in graph drawing (to be fair, we should mention that it has appeared before in one work by Shin et al. [26], on poly-line orthogonal drawings of binary trees with $O(1)$ bends per edge). We realize that more can be gained from the recursion in the previous $O(n \log\log n)$ algorithm, by bootstrapping. This requires a careful setup of the recursive subproblems, and constant switching of x and y (width and height) every time we bootstrap. (The author is reminded of an algorithm by Matoušek [21] on a completely different problem, Hopcroft's problem, where iterated logarithm arose due to constant switching of points and lines by duality at each level of recursion.)

Our $n2^{O(\sqrt{\log n})}$ result for orthogonal order-preserving drawings has the largest quantitative improvement compared to previous results, but actually requires the least originality in techniques. We use the exact same form of recursion as in an earlier algorithm of Chan [6] for non-orthogonal upward order-preserving drawings, although the new algorithm requires trickier details.

2 Straight-line upward drawings of arbitrary trees

In this section, we consider arbitrary (rooted) trees and describe our first algorithm to produce straight-line upward drawings with $o(n \log n)$ area. It serves as a warm-up to the further improved algorithm in Section 3 when upwardness is dropped.

2.1 Preliminaries

We begin with some basic number-theoretic and tree-drawing facts. The first, on the denseness of relatively prime pairs, is well known:

▶ **Fact 1.** *There are $\Omega(AB)$ relatively prime pairs in $\{1, \ldots, A\} \times \{\lfloor B/2 \rfloor + 1, \ldots, B\}$.*

Next, we consider drawing trees not on the integer grid but on a user-specified set of points. We note that any point set of near linear size that is not too degenerate is "universal", in the sense that it can be used to draw any tree.

▶ **Fact 2.** *Let P be a set of $(\ell - 1)n - \ell + 2$ points in the plane, with no ℓ points lying on a common line. Let T be a tree of size n. Then T has a straight-line upward drawing where all vertices are drawn in P.*

Proof. We describe a straightforward recursive algorithm: Let $n_1, \ldots, n_d$ be the sizes of the subtrees $T_1, \ldots, T_d$ at the children of the root v_0, with $\sum_{i=1}^{d} n_i = n - 1$. Place v_0 at the highest point p_0 of P (in case of ties, prefer the leftmost highest point). Form d disjoint sectors with apex at p_0, so that the i-th sector S_i contains between $(\ell - 1)n_i - \ell + 2$ and $(\ell - 1)n_i$ points of $P - \{p_0\}$. This is possible since any line through p_0 contains at most $\ell - 2$ points of $P - \{p_0\}$, and $\sum_{i=1}^{d} (\ell - 1)n_i = (\ell - 1)(n - 1) = |P - \{p_0\}|$. For each $i = 1, \ldots, d$, recursively draw T_i using $(\ell - 1)n_i - \ell + 2$ points of $P \cap S_i$. Lastly, draw the edges from

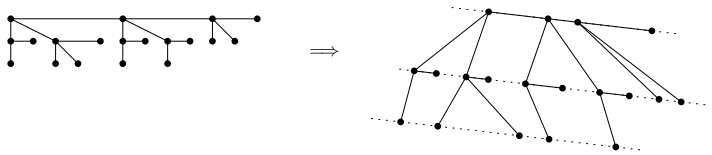


Figure 2 The drawing in Fact 3.

v_0 to the roots of the T_i's (these edges create no crossings since the roots are drawn at the highest points of P in their respective sectors). The base case $n = 1$ is trivial. ◀

The following is a slight generalization of the standard algorithm (mentioned in the introduction) for straight-line upward drawings of general trees with width $O(n)$ and height $O(\log n)$. We note that the algorithm can draw any tree on any point set that "behaves" like an $n \times \lceil \log n \rceil$ grid.

▶ **Fact 3.** *Let G be a set of $\lceil \log n \rceil$ parallel (non-vertical) line segments in the plane. Let P be a set of $n \lceil \log n \rceil$ points, with n points lying on each of the $\lceil \log n \rceil$ line segments in G. Let T be a tree of size n. Then T has a straight-line drawing where all vertices are drawn in P, and the root is drawn on the segment of G whose line has the highest y-intercept.*

Furthermore, if the segments of G are horizontally separated (i.e., the y-projections are disjoint), the drawing is upward.

Proof. Without loss of generality, assume that the segments have negative slope, and arrange the segments of G in decreasing order of y-intercepts. Apply the standard algorithm to get a straight-line upward grid drawing of T with width at most n and height at most $\lceil \log n \rceil$. Map the vertices on the i-th topmost row of the grid drawing to the points on the i-th segment of G, while preserving the left-to-right ordering of the vertices. (See Figure 2.) The resulting drawing is planar (since each edge is drawn either on a segment or in the region between two consecutive segments, and there are no crossings in the region between two consecutive segments). Note that the drawing is upward if the segments of G are horizontally separated. ◀

2.2 The augmented-star algorithm

The main difficulty of drawing arbitrary trees is due to the presence of vertices of large degree. In the extreme case when the tree is a star of size n, we can produce a straight-line drawing of width $O(A)$ and $O(n/A)$ for any given $1 \leq A \leq n$, by placing the root at the origin and placing the remaining vertices at points with co-prime x- and y-coordinates, using Fact 1.

We first study a slightly more general special case which we call *augmented stars*, where the input tree is modified from a star by attaching to each leaf a small subtree of size $\leq s$.

▶ **Lemma 4.** *Let T be a tree of size n such that the subtree at each child of the root has size at most s. For any given $n \geq A \geq 1$, T has a straight-line upward drawing with width $O(A \log s)$ and height $O((n/A) \cdot s \log^2 s)$, where the root is placed at the top left corner of the bounding box, and the left side of the box contains no other vertices.*

Proof. Let $\ell = s \lceil \log s \rceil$. Let $B = \lceil c\ell n/A \rceil$ for some constant c. Let $P = \{(x, y) \in \{1, \ldots, A\} \times \{-B, \ldots, -\lfloor B/2 \rfloor - 1\} : x \text{ and } y \text{ are relatively prime}\}$. By Fact 1, $|P| = \Omega(AB)$, and so $|P| \geq \ell n$ by making c sufficiently large.

Let $n_1, \ldots, n_d$ be the sizes of the subtrees $T_1, \ldots, T_d$ at the children of the root v_0, with $\sum_{i=1}^{d} n_i = n - 1$ and $n_i \leq s$ for each i. Place v_0 at the origin. Form d disjoint sectors, where

the i-th sector S_i contains exactly ℓn_i points of P. This is possible, since any line through the origin contains at most one point of P and $\sum_{i=1}^{d} \ell n_i < \ell n \le |P|$. We will draw T using not just the points of P, but also scaled copies of these points, up to scaling factor $t := \lceil \log s \rceil$.

For each i, consider two cases, depending on how degenerate $S_i \cap P$ is:

- Case 1: S_i does not contain ℓ points of P on a common line. Here, we can draw T_i using the $\ell n_i > (\ell - 1)n_i - \ell + 2$ points of $S_i \cap P$ by Fact 2.
- Case 2: S_i contains ℓ points of P on a common line L. (Note that L does not pass through the origin, by definition of P.) Let σ be a horizontal slab of height $B/(2t)$ that contains at least $\ell/t = s$ points of $L \cap S_i \cap P$. Let $\overline{L} = L \cap S_i \cap \sigma$. Let G be the set of t line segments $\overline{L}, 2\overline{L}, \ldots, t\overline{L}$, where $\alpha\overline{L}$ denotes the scaled copy of $\overline{L}$ by factor α (with respect to the origin). Each of the $t = \lceil \log s \rceil$ segments of G contain s integer points inside S_i, and the segments are horizontally separated. Thus, we can draw T_i using the integer points on G by Fact 3.

Lastly, draw the edges from v_0 to the roots of the T_i's. The total width is $O(tA) = O(A \log s)$ and the height is $O(tB) = O((n/A) \cdot s \log^2 s)$. ◀

2.3 The general algorithm

We are now ready to present the algorithm for the general case, using the augmented-star algorithm as a subroutine:

▶ **Theorem 5.** *For any given $n \ge A \ge 1$, every tree T of size n has a straight-line upward drawing with width $O(A + \log n)$ and height $O((n/\sqrt{A}) \log^2 A)$, where the root is placed at the top left corner of the bounding box.*

Proof. We describe a recursive algorithm to draw T: Let s be a fixed parameter with $A \ge \log s$. Let v_0 be the root of T, and define v_{i+1} to be the child of v_i whose subtree is the largest (the resulting root-to-leaf path $v_0 v_1 v_2 \cdots$ is called the *heavy path* of T). Let k be the largest index such that the subtree at v_k has size more than $n - A$ (we will call the node v_k the *A-skewed centroid*). Then the total size of the subtrees at the siblings of $v_1, \ldots, v_k$ is at most A, the subtree at v_{k+1} has size at most $n - A$, and the subtree at each sibling of v_{k+1} has size at most $\min\{n - A, n/2\}$.

The drawing of T, depicted in Figure 3, is constructed as follows (which includes multiple applications of the standard algorithm in steps 1 and 3, one application of the augmented-star algorithm in step 2, and recursive calls in step 4):

1. Draw the subtrees at the siblings of $v_1, \ldots, v_k$ by the standard algorithm. Stack these drawings horizontally. Since these subtrees have total size at most A, the drawing so far has total width $O(A)$ and height $O(\log A)$.
2. Draw the subtrees at the children of v_k that have *size* $\le s$, together with the edges from v_k to the roots of these subtrees, by the augmented-star algorithm in Lemma 4 with parameter $\tilde{A} = \lceil A / \log s \rceil$. By reflection, make v_k lie on the top-right corner of its corresponding bounding box. Place the drawing below the drawings from step 1. This part has width $O(\tilde{A} \log s) = O(A)$ and height $O((n'/\tilde{A}) \cdot s \log^2 s) = O((n'/A) \cdot s \log^3 s)$ where n' is the total size of these subtrees.
   (Note that if $n' \le A$, we can just use the standard algorithm with width $O(A)$ and height $O(\log A)$ for this step.)
3. Draw the subtrees at the children of v_k that have *size* $> s$ *and* $\le A$, by the standard algorithm. By reflection, make the roots lie on the top-right corners of their respective bounding boxes. Stack these drawings vertically, underneath the drawing from step 2. This

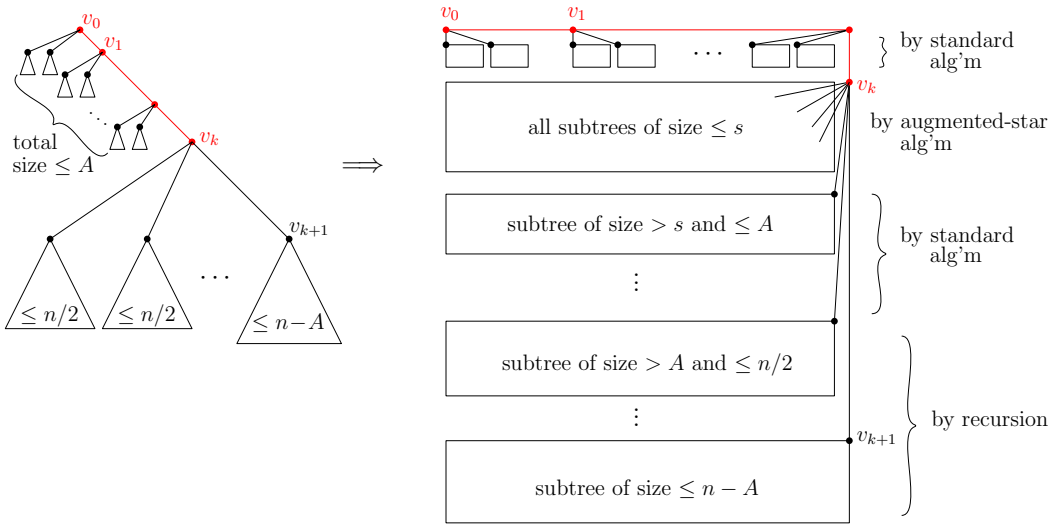


Figure 3 The general algorithm in Theorem 5.

part has width $O(A)$ and height $O((\text{number of these subtrees}) \cdot \log A) \leq O((n''/s) \cdot \log A)$, where n'' is the total size of these subtrees.

4. Recursively draw the subtrees at the children of v_k that have *size* $> A$. By reflection, make the roots lie on the top-right corners of their respective bounding boxes. Stack these drawings vertically, underneath the drawings from step 3. Put the drawing of the subtree at v_{k+1} at the bottom.

The special case $k = 1$ is similar, except that we place v_k on the left, and so do not reflect in steps 2–4. The special case $k = 0$ is also similar, but bypassing step 1.

The overall width satisfies the following recurrence $W(n) \leq \max\{O(A),\ W(n/2) + 1,\ W(n - A)\}$, which solves to $W(n) = O(A + \log n)$.

The overall height satisfies the following recurrence

$$H(n) \leq \sum_{i=1}^{m} H(n_i) + c(\log A + (n'/A)s \log^3 s + (n''/s) \log A)$$

for some $n', n'', m, n_1, \ldots, n_m$ with $n' + n'' + \sum_i n_i \leq n$, $n_i \leq n - A$, and $n_i \geq A$, for some constant c. It is straightforward to verify by induction that $H(n) \leq c((2n/A - 1) \log A + (n/A)s \log^3 s + (n/s) \log A)$. (The constraint $n_i \leq n - A$ is needed in the $m = 1$ case.) Choosing $s = \Theta(\sqrt{A}/\log A)$ to balance the last two terms gives the height bound in the theorem. ◀

Finally, choosing $A = \lceil \log n \rceil$ gives area $O(n\sqrt{\log n} \log^2 \log n)$.

3 Straight-line drawings of arbitrary trees

To obtain still better area bounds for straight-line non-upward drawings of arbitrary trees, the idea is to bootstrap: we show how to use a given general algorithm to obtain an improved augmented-star algorithm, which in turn is used to obtain an improved general algorithm. In order to bootstrap, we need to identify large grid substructures inside each sector in the augmented-star algorithm. This requires an interesting geometric observation about lattices, described in the following subsection.

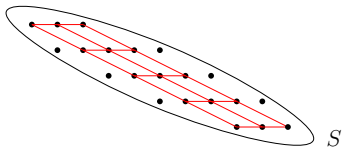


Figure 4 Observation 6: A convex set that contains many lattice points must contain a large affine grid in the lattice.

3.1 An observation about lattices

A *two-dimensional lattice* is a set of the form $\Lambda = \{i\mathbf{u} + j\mathbf{v} : i, j \in \mathbb{Z}\}$ for some vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^2$. The vector pair $\{\mathbf{u}, \mathbf{v}\}$ is called a *basis* of Λ.

In this paper, we use the term $a \times b$ *affine grid* to refer to a set of the form $\{i\mathbf{u} + j\mathbf{v} : i \in \{x_0+1, \ldots, x_0+a\}, j \in \{y_0+1, \ldots, y_0+b\}\}$ for some vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^2$ and some $x_0, y_0 \in \mathbb{R}$. In other words, it is a set that is equivalent to the regular $a \times b$ grid $\{1, \ldots, a\} \times \{1, \ldots, b\}$ after applying some affine transformation.

The following observation is the key (see Figure 4). The author is not aware of any references of this specific statement (but would not be surprised if this was known before).

▶ **Observation 6.** *If a convex set S in the plane contains n points from a lattice Λ, then $S \cap \Lambda$ contains an $a \times b$ affine grid for some a and b with $ab = \Omega(n)$.*

Proof. First, apply an affine transformation to make S *fat*, i.e., $D^- \subset S \subset D^+$ for some disks D^- and D^+ with $\mathrm{diam}(D^-) = \Omega(\mathrm{diam}(D^+))$. (This follows immediately from well-known properties of the *Löwner–John ellipsoid*; or see [1, 3] for simple, direct algorithms.)

After the transformation, Λ is still a lattice. It is well known that there exists a basis $\{\mathbf{u}, \mathbf{v}\}$ for Λ satisfying $60° \le \angle(\mathbf{u}, \mathbf{v}) \le 120°$. (A *Gauss-reduced basis* satisfies this property; for example, see [29, Section 27.2].)

Let R^+ be the smallest rhombus containing D^+, with sides parallel to $\mathbf{u}$ and $\mathbf{v}$. Let R^- be the largest rhombus R^- contained in D^-, with sides parallel to $\mathbf{u}$ and $\mathbf{v}$. Then R^+ and R^- have side lengths $r^+ = O(\mathrm{diam}(D^+))$ and $r^- = \Omega(\mathrm{diam}(D^-))$ respectively, since $\angle(\mathbf{u}, \mathbf{v})$ is bounded away from 0° or 180°. It follows that $r^- = \Omega(r^+)$.

Now, $S \cap \Lambda \subset R^+ \cap \Lambda$ is contained in an $\lceil r^+/\|\mathbf{u}\| \rceil \times \lceil r^+/\|\mathbf{v}\| \rceil$ affine grid. Thus, $n \le \lceil r^+/\|\mathbf{u}\| \rceil \cdot \lceil r^+/\|\mathbf{v}\| \rceil$.

On the other hand, $S \cap \Lambda \supset R^- \cap \Lambda$ contains an $\lfloor r^-/\|\mathbf{u}\| \rfloor \times \lfloor r^-/\|\mathbf{v}\| \rfloor$ affine grid, with $\lfloor r^-/\|\mathbf{u}\| \rfloor \times \lfloor r^-/\|\mathbf{v}\| \rfloor = \Omega(\lceil r^+/\|\mathbf{u}\| \rceil \cdot \lceil r^+/\|\mathbf{v}\| \rceil) = \Omega(n)$ points, assuming that $\|\mathbf{u}\|, \|\mathbf{v}\| \le r^-$.

This almost completes the proof. It remains to address the special case when $\|\mathbf{u}\| > r^-$ (the case $\|\mathbf{v}\| > r^-$ is similar). Here, $S \cap \Lambda \subset R^+ \cap \Lambda$ is contained in an $O(1) \times \lceil r^+/\|\mathbf{v}\| \rceil$ affine grid. Some row of the grid must contain $\Omega(n)$ points of $S \cap \Lambda$. The row is a $1 \times \Omega(n)$ affine grid. ◀

3.2 Improved augmented-star algorithm

We first show how to use a given general algorithm $\mathcal{G}_0$ to obtain an improved algorithm for the augmented-star case:

▶ **Lemma 7.** *Suppose we are given a* general algorithm $\mathcal{G}_0$ *that takes as input any $n \ge A \ge g_0(n)$ and any tree of size n, and outputs a straight-line drawing of width at most A and height at most $(n/A)f_0(A)$, where the root is drawn at the top left corner of the bounding box. Here, f_0 and g_0 are some increasing functions satisfying $f_0(n) \ge g_0(n)$.*

Then we can obtain an improved augmented-star algorithm *that takes as input any $n \geq A \geq 1$ and a tree of size n such that the subtree at each child of the root has size at most s, and outputs a straight-line drawing with width $O(A \log s)$ and height $O((n/A) \cdot f_0(s) \log s)$, where the root is placed at the top left corner of the bounding box, and the left side of the box contains no other vertices.*

Proof. Let $\ell = cf_0(s)$ for some constant c. Let $B = \lceil c\ell n/A \rceil$. Let $P = \{(x, y) \in \{1, \ldots, A\} \times \{-B, \ldots, -1\} : x \text{ and } y \text{ are relatively prime}\}$. By Fact 1, $|P| = \Omega(AB)$, and so $|P| \geq \ell n$ by making c sufficiently large.

Let $n_1, \ldots, n_d$ be the sizes of the subtrees $T_1, \ldots, T_d$ at the children of the root v_0, with $\sum_{i=1}^{d} n_i = n - 1$ and $n_i \leq s$ for each i. Place v_0 at the origin. Form d disjoint sectors, where the i-th sector S_i contains exactly ℓn_i points of P. This is possible, since any line through the origin contains at most one point of P and $\sum_{i=1}^{d} \ell n_i < \ell n \leq |P|$.

Take a fixed i. Applying Observation 6 to the convex set $S_i \cap ((0, A] \times [-B, 0))$, we see that $S_i \cap (\{1, \ldots, A\} \times \{-B, \ldots, -1\})$ must contain an $a \times b$ affine grid for some a and b with $ab = \Omega(\ell n_i)$. Note that $b \geq (n_i/a) f_0(s)$ by making c sufficiently large. Consider two cases:

- Case 1: $g_0(n_i) \leq a \leq n_i$. Here, we can draw T_i in the $a \times b$ affine grid by the given algorithm $\mathcal{G}_0$, after applying an affine transformation to convert to a standard integer $a \times b$ grid. Note that planarity and straightness are preserved under the transformation (but not upwardness). The root of T_i can be placed at the highest corner of the grid.
- Case 2: $a > n_i$ or $a < g_0(n_i)$. Note that in the latter subcase, $b \geq (n_i/a) f_0(s) \geq (n_i/a) g_0(n_i) \geq n_i$. In either subcase, S_i contains n_i points of P on a common line L. (Note that L does not pass through the origin, by definition of P.) Let $t = \lceil \log s \rceil$ and $\overline{L} = L \cap S_i$. Let G be the t line segments $\overline{L}, 2\overline{L}, \ldots, t\overline{L}$. Then each of the $t = \lceil \log s \rceil$ segments of G contain n_i integer points inside S_i. Thus, we can draw T_i using the integer points on G by Fact 3. The root is placed on the highest segment of G.

Lastly, draw the edges from v_0 to the roots of the T_i's. The total width is $O(tA) = O(A \log s)$ and height is $O(tB) = O((n/A) \cdot f_0(s) \log s)$. ◀

3.3 Improved general algorithm

Using the improved augmented-star algorithm, we can then obtain an improved general algorithm, by following the same approach as in the proof of Theorem 5, except with Lemma 4 replaced by the improved Lemma 7 in step 2. The same analysis shows the following:

▶ **Theorem 8.** *Suppose we are given a* general algorithm $\mathcal{G}_0$ *that takes as input any $n \geq A \geq g_0(n)$ and any tree of size n, and outputs a straight-line drawing of width at most A and height at most $(n/A) f_0(A)$, where the root is drawn at the top left corner of the bounding box. Here, f_0 and g_0 are some increasing functions satisfying $f_0(n) \geq g_0(n)$.*

Then we can obtain an improved general algorithm *that takes as input any $n \geq A \geq \log s$ and any tree of size n, and outputs a straight-line upward drawing with width $O(A + \log n)$ and height $O((n/A) \log A + (n/A) f_0(s) \log^2 s + (n/s) \log A)$, where the root is placed at the top left corner of the bounding box.*

Assume inductively that there is a general algorithm $\mathcal{G}_0$ satisfying the assumption of the above theorem with $f_0(A) = C_j A^{1/j} \log^j A$ and $g_0(n) = c_0 \log n$ for some C_j and c_0. For $j = 1$, this follows from the standard algorithm, which has logarithmic width and linear height after swapping x and y, with $C_1, c_0 = O(1)$.

Choosing $s = \lceil A^{j/(j+1)}/\log^j A \rceil$ to balance the last two terms in the above theorem gives a width bound of $O(A + \log n)$ and height bound of

$$O((n/A)\log A + (n/A)C_j s^{1/j}\log^{j+2} s + (n/s)\log A) \;=\; O(C_j(n/A)A^{1/(j+1)}\log^{j+1} A).$$

By setting $\tilde{A} = c_0 A$ and $C_{j+1} = O(1)\cdot C_j$, with a sufficiently large absolute constant c_0, the width is at most $\tilde{A}$ and the height is at most $C_{j+1}(n/\tilde{A})\tilde{A}^{1/(j+1)}\log^{j+1}\tilde{A}$ for any $n \geq \tilde{A} \geq c_0 \log n$. We have thus obtained a new general algorithm with $f_0(\tilde{A}) = C_{j+1}\tilde{A}^{1/(j+1)}\log^{j+1}\tilde{A}$ and $g_0(n) = c_0 \log n$.

Note that $C_j = 2^{O(j)}$. For the best bound, we choose a nonconstant $j = \Theta(\sqrt{\log A/\log\log A})$ so that $f_0(A) = 2^{O(j)}A^{1/j}\log^j A = 2^{O((\log A)/j + j\log\log A)} = 2^{O(\sqrt{\log A \log\log A})}$, yielding:

▶ **Corollary 9.** *For any given $n \geq A \geq \log n$, every tree of size n has a straight-line drawing with width $O(A)$ and height $(n/A)2^{O(\sqrt{\log A \log\log A})}$.*

Finally, choosing $A = \lceil \log n \rceil$ gives area $n2^{O(\sqrt{\log\log n \log\log\log n})}$.

▶ Remark. It is straightforward to implement the algorithms in Section 2 and this section in polynomial time. One open question is whether the improved bound holds for upward drawings. Another open question is whether further improvements are possible if we allow poly-line drawings.

4 Straight-line orthogonal drawings of binary trees

In this section, we consider binary trees and describe algorithms to produce straight-line orthogonal (non-upward) drawings. We improve previous algorithms with $O(n\log\log n)$ area by Shin, Kim, and Chwa [25] and Chan et al. [7]. The idea is (again) to bootstrap.

Given a binary tree T and two distinct vertices u and v, such that v is a descendant of u but not an immediate child of v, the *chain* from u to v is defined to be the subtree at u minus the subtree at v. (To explain the terminology, note that the chain consists of the path from u to the parent of v, together with a sequence of subtrees attached to the nodes of this path.) We show how to use a given algorithm for drawing chains to obtain a general algorithm for drawing trees, which together with the given chain algorithm is used to obtain an improved chain algorithm.

4.1 The general algorithm

Given a chain algorithm $\mathcal{C}_0$, we can naively use it to draw the entire tree, since a tree can be viewed as a chain from the root to an artificially created leaf. We first show how to use a given chain algorithm $\mathcal{C}_0$ to obtain a general algorithm that achieves *arbitrary width–height tradeoffs*. This is done by adapting previous algorithms [25, 7].

▶ **Lemma 10.** *Suppose we are given a* chain algorithm $\mathcal{C}_0$ *that takes as input any binary tree and a chain from v_0 to v_k where the size of the chain is n, and outputs a straight-line orthogonal drawing of the chain with width at most $W_0(n)$ and height at most $H_0(n)$, where v_0 is placed at the top left corner of the bounding box, and the parent of v_k is placed at the bottom left corner of the box. Here, $W_0(n)$ and $H_0(n)$ are increasing functions.*

Then we can obtain a general algorithm *that takes as input $n \geq A \geq 1$ and any binary tree T of size n, and outputs a straight-line orthogonal drawing with width $O(W_0(A) + \log n)$ and height $O((n/A)H_0(A))$, where the root is placed at the top left corner of the bounding box.*

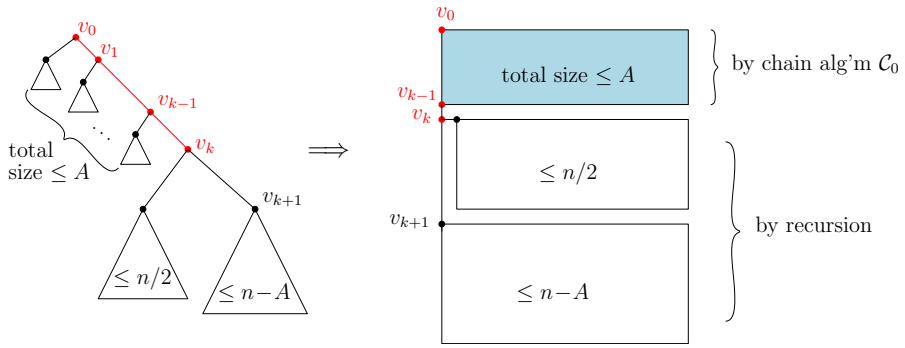


Figure 5 The general algorithm in Lemma 10 for orthogonal drawings.

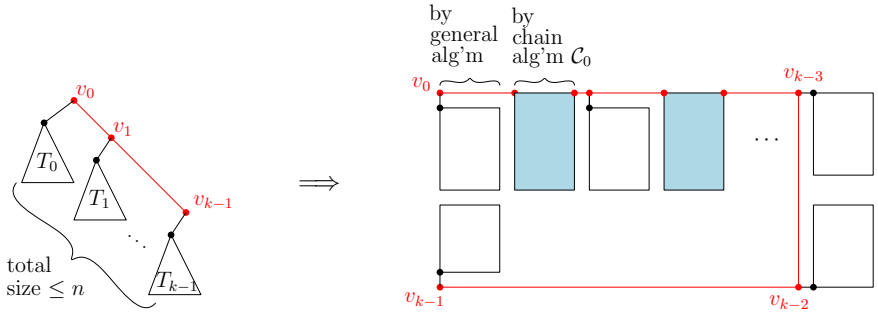


Figure 6 The improved chain algorithm in Theorem 11 for orthogonal drawings.

Proof. We describe a recursive algorithm to draw T: Let $v_0v_1v_2\cdots$ be the heavy path, and v_k be the A-skewed centroid, as in the proof of Theorem 5. Then the chain from v_0 to v_k has size at most A, the subtree at v_{k+1} has size at most $n-A$, and the subtree at the sibling of v_{k+1} has size at most $\min\{n-A, n/2\}$.

The drawing of T, depicted in Figure 5, is constructed as follows:

1. Draw the chain from v_0 to v_k by the given algorithm $\mathcal{C}_0$, with width at most $W_0(A)$ and height at most $H_0(A)$.
2. Recursively draw the subtrees at the two children of v_k. Stack the two drawings vertically, underneath the drawing from step 1. Put the drawing of the subtree at v_{k+1} at the bottom. (Note that if any of these subtrees has size at most A, we can just use algorithm $\mathcal{C}_0$ with width at most $W_0(A)$ and height at most $H_0(A)$.)

The special case $k=1$ is similar, except that in step 1 we can just apply algorithm $\mathcal{C}_0$ to draw the subtree at the sibling of v_1, and connect v_0 to v_k directly. The special case $k=0$ is also similar, but bypassing step 1.

The overall width satisfies the recurrence $W(n) \le \max\{O(W_0(A)),\ W(n/2)+1,\ W(n-A)\}$, which solves to $W(n) = O(W_0(A)+\log n)$.

The overall height satisfies the recurrence $H(n) \le \sum_{i=1}^{m} H(n_i) + cH_0(A)$ for some $m, n_1, \ldots, n_m$ with $m \le 2$, $\sum_i n_i \le n$, $n_i \le n-A$, and $n_i \ge A$, for some constant c. The recurrence solves to $H(n) \le c(2n/A-1)H_0(A)$ (similarly to the proof of Theorem 5). ◀

4.2 The improved chain algorithm

Using both the general algorithm from Lemma 10 and the given chain algorithm $\mathcal{C}_0$, we describe an improved chain algorithm:

▶ **Theorem 11.** *Suppose we are given a* chain algorithm $\mathcal{C}_0$ *that takes as input any binary tree and a chain from v_0 to v_k where the size of the chain is n, and outputs a straight-line orthogonal drawing of the chain with width at most $W_0(n)$ and height at most $H_0(n)$, where v_0 is placed at the top left corner of the bounding box, and the parent of v_k is placed at the bottom left corner of the box. Here, $W_0(n)$ and $H_0(n)$ are increasing functions.*

Then we can obtain an improved chain algorithm *that takes as input any $n \geq A \geq 1$ and any binary tree and a chain from v_0 to v_k where the size of the chain is n, and outputs a straight-line orthogonal drawing of the chain with width $O((n/A)H_0(A))$ and height $O(W_0(A) + \log n)$, where v_0 is placed at the top left corner of the bounding box, and the parent of v_k is placed at the bottom left corner.*

Proof. Let $v_0v_1 \cdots v_k$ denote the path from v_0 to v_k. Let T_i denote the subtree at the sibling of v_{i+1}. Let n_i be the size of T_i plus 1.

Divide the sequence $v_0v_1 \cdots v_{k-4}$ into subsequences, where each subsequence is either (i) a *singleton* v_i, or (ii) a contiguous *block* $v_iv_{i+1} \cdots v_\ell$ of length at least 2 with $n_i+n_{i+1}+\cdots+n_\ell \leq A$. By making the blocks maximal, we can ensure that the number of singletons and blocks is $O(n/A)$. We add $v_{k-3}, \ldots, v_{k-1}$ as 3 extra singletons.

- For each singleton v_i, draw T_i by the general algorithm in Lemma 10 if $n_i \geq A$, or directly by the given algorithm $\mathcal{C}_0$ if $n_i < A$. By swapping x and y, the width is $O((n_i/A + 1)H_0(A))$ and the height is $O(W_0(A) + \log n)$.
- For each block $v_iv_{i+1} \cdots v_\ell$, draw the subchain from v_i to $v_{\ell+1}$, which has size at most A, by the given algorithm $\mathcal{C}_0$. By swapping x and y, the width is $O(H_0(A))$ and the height is $O(W_0(A))$.

All these drawings are stacked horizontally as shown in Figure 6, except for T_{k-2} and T_{k-1}, which are placed below and flipped upside-down.

The special cases with $k \leq 3$ are simpler: just stack the $O(1)$ drawings vertically, with the bottom drawing of T_{k-1} flipped upside-down.

The total width due to singletons is $O(\sum_i (n_i/A + 1)H_0(A)) = O((n/A)H_0(A))$, and the total width due to blocks is also $O((n/A)H_0(A))$, because the number of singletons and blocks is $O(n/A)$. The overall height is $O(W_0(A) + \log n)$. ◀

Assume inductively that there is a chain algorithm $\mathcal{C}_0$ satisfying the assumption of Theorem 11 with $W_0(n) = C_j(n/\log n)\log^{(j)} n$ and $H_0(n) = C_j \log n$ for some C_j, where $\log^{(j)}$ denotes the j-th iterated logarithm. For $j = 1$, this follows by simply applying the standard algorithm to draw the subtrees T_i in the proof of Theorem 11, with $C_1 = O(1)$.

Choosing $A = \left\lceil \log n \log\log n / \log^{(j+1)} n \right\rceil$ in Theorem 11 gives a width bound of $O((n/A)H_0(A)) = O((n/A)C_j \log A) = O(C_j(n/\log n)\log^{(j+1)} n)$ and a height bound of $O(W_0(A) + \log n) = O(C_j(A/\log A)\log^{(j)} A + \log n) = O(C_j \log n)$. By setting $C_{j+1} = O(1)\cdot C_j$, we have thus obtained a new chain algorithm with $W_0(n) = C_{j+1}(n/\log n)\log^{(j+1)} n$ and $H_0(n) = C_{j+1} \log n$.

Note that $C_j = 2^{O(j)}$. For the best bound, we choose a nonconstant $j = \log^* n$, yielding:

▶ **Corollary 12.** *Every binary tree of size n has a straight-line orthogonal drawing with area $n2^{O(\log^* n)}$.*

References

1 Pankaj K. Agarwal, Sariel Har-Peled, and Kasturi R. Varadarajan. Approximating extent measures of points. *J. ACM*, 51(4):606–635, 2004. doi:10.1145/1008731.1008736.

2 Christian Bachmaier, Franz-Josef Brandenburg, Wolfgang Brunner, Andreas Hofmeier, Marco Matzeder, and Thomas Unfried. Tree drawings on the hexagonal grid. In *Proc. 16th International Symposium on Graph Drawing (GD)*, pages 372–383, 2008. doi:10.1007/978-3-642-00219-9_36.

3 Gill Barequet and Sariel Har-Peled. Efficiently approximating the minimum-volume bounding box of a point set in three dimensions. *J. Algorithms*, 38(1):91–109, 2001. doi:10.1006/jagm.2000.1127.

4 Therese Biedl. Ideal drawings of rooted trees with approximately optimal width. *J. Graph Algorithms Appl.*, 21:631–648, 2017. doi:10.7155/jgaa.00432.

5 Therese Biedl. Upward order-preserving 8-grid-drawings of binary trees. In *Proc. 29th Canadian Conference on Computational Geometry (CCCG)*, pages 232–237, 2017. URL: http://2017.cccg.ca/proceedings/Session6B-paper4.pdf.

6 Timothy M. Chan. A near-linear area bound for drawing binary trees. *Algorithmica*, 34(1):1–13, 2002. doi:10.1007/s00453-002-0937-x.

7 Timothy M. Chan, Michael T. Goodrich, S. Rao Kosaraju, and Roberto Tamassia. Optimizing area and aspect ration in straight-line orthogonal tree drawings. *Comput. Geom.*, 23(2):153–162, 2002. doi:10.1016/S0925-7721(01)00066-9.

8 Pierluigi Crescenzi, Giuseppe Di Battista, and Adolfo Piperno. A note on optimal area algorithms for upward drawings of binary trees. *Comput. Geom.*, 2:187–200, 1992. doi:10.1016/0925-7721(92)90021-J.

9 Pierluigi Crescenzi and Paolo Penna. Strictly-upward drawings of ordered search trees. *Theor. Comput. Sci.*, 203(1):51–67, 1998. doi:10.1016/S0304-3975(97)00287-9.

10 Hubert de Fraysseix, János Pach, and Richard Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10(1):41–51, 1990. doi:10.1007/BF02122694.

11 Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing*. Prentice Hall, 1999.

12 Giuseppe Di Battista and Fabrizio Frati. A survey on small-area planar graph drawing. *CoRR*, abs/1410.1006, 2014. arXiv:1410.1006.

13 Fabrizio Frati. Straight-line orthogonal drawings of binary and ternary trees. In *Proc. 15th International Symposium on Graph Drawing (GD)*, pages 76–87, 2007. doi:10.1007/978-3-540-77537-9_11.

14 Fabrizio Frati, Maurizio Patrignani, and Vincenzo Roselli. LR-drawings of ordered rooted binary trees and near-linear area drawings of outerplanar graphs. In *Proc. 28th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1980–1999, 2017. doi:10.1137/1.9781611974782.129.

15 Ashim Garg, Michael T. Goodrich, and Roberto Tamassia. Planar upward tree drawings with optimal area. *Int. J. Comput. Geometry Appl.*, 6(3):333–356, 1996. Preliminary version in SoCG'93. doi:10.1142/S0218195996000228.

16 Ashim Garg and Adrian Rusu. Area-efficient order-preserving planar straight-line drawings of ordered trees. *Int. J. Comput. Geometry Appl.*, 13(6):487–505, 2003. doi:10.1142/S021819590300130X.

17 Ashim Garg and Adrian Rusu. Straight-line drawings of general trees with linear area and arbitrary aspect ratio. In *Proc. 3rd International Conference on Computational Science and Its Applications (ICCSA), Part III*, pages 876–885, 2003. doi:10.1007/3-540-44842-X_89.

18 Ashim Garg and Adrian Rusu. Straight-line drawings of binary trees with linear area and arbitrary aspect ratio. *J. Graph Algorithms Appl.*, 8(2):135–160, 2004. URL: http://jgaa.info/accepted/2004/GargRusu2004.8.2.pdf.

19 Stephanie Lee. Upward octagonal drawings of ternary trees. Master's thesis, University of Waterloo, 2016. (Supervised by T. Biedl and T. M. Chan.) https://uwspace.uwaterloo.ca/handle/10012/10832.

20 Charles E. Leiserson. Area-efficient graph layouts (for VLSI). In *Proc. 21st IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 270–281, 1980. doi:10.1109/SFCS.1980.13.

21 Jiří Matoušek. Range searching with efficient hiearchical cutting. *Discrete & Computational Geometry*, 10:157–182, 1993. doi:10.1007/BF02573972.

22 Edward M. Reingold and John S. Tilford. Tidier drawings of trees. *IEEE Trans. Software Eng.*, 7(2):223–228, 1981. doi:10.1109/TSE.1981.234519.

23 Walter Schnyder. Embedding planar graphs on the grid. In *Proc. 1st ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 138–148, 1990. URL: http://dl.acm.org/citation.cfm?id=320176.320191.

24 Yossi Shiloach. *Linear and Planar Arrangement of Graphs*. PhD thesis, Weizmann Institute of Science, 1976. URL: https://lib-phds1.weizmann.ac.il/Dissertations/shiloach_yossi.pdf.

25 Chan-Su Shin, Sung Kwon Kim, and Kyung-Yong Chwa. Area-efficient algorithms for straight-line tree drawings. *Comput. Geom.*, 15(4):175–202, 2000. Preliminary version in COCOON'96. doi:10.1016/S0925-7721(99)00053-X.

26 Chan-Su Shin, Sung Kwon Kim, Sung-Ho Kim, and Kyung-Yong Chwa. Algorithms for drawing binary trees in the plane. *Inf. Process. Lett.*, 66(3):133–139, 1998. doi:10.1016/S0020-0190(98)00049-0.

27 Luca Trevisan. A note on minimum-area upward drawing of complete and fibonacci trees. *Inf. Process. Lett.*, 57(5):231–236, 1996. doi:10.1016/0020-0190(96)81422-0.

28 Leslie G. Valiant. Universality considerations in VLSI circuits. *IEEE Trans. Computers*, 30(2):135–140, 1981. doi:10.1109/TC.1981.6312176.

29 Vijay V. Vazirani. *Approximation Algorithms*. Springer, 2001. URL: https://www.cc.gatech.edu/fac/Vijay.Vazirani/book.pdf.

Approximate Shortest Paths and Distance Oracles in Weighted Unit-Disk Graphs

Timothy M. Chan
Department of Computer Science, University of Illinois at Urbana-Champaign, USA
tmc@illinois.edu

Dimitrios Skrepetos
Cheriton School of Computer Science, University of Waterloo, Canada
dskrepet@uwaterloo.ca

Abstract

We present the first near-linear-time $(1+\varepsilon)$-approximation algorithm for the *diameter* of a weighted unit-disk graph of n vertices, running in $O\left(n \log^2 n\right)$ time, for any constant $\varepsilon > 0$, improving the near-$O\left(n^{3/2}\right)$-time algorithm of Gao and Zhang [STOC 2003]. Using similar ideas, we can construct a $(1+\varepsilon)$-approximate *distance oracle* for weighted unit-disk graphs with $O(1)$ query time, with a similar improvement in the preprocessing time, from near $O\left(n^{3/2}\right)$ to $O\left(n \log^3 n\right)$. We also obtain new results for a number of other related problems in the weighted unit-disk graph metric, such as the radius and bichromatic closest pair.

As a further application, we use our new distance oracle, along with additional ideas, to solve the $(1+\varepsilon)$-approximate *all-pairs bounded-leg shortest paths* problem for a set of n planar points, with near $O\left(n^{2.579}\right)$ preprocessing time, $O\left(n^2 \log n\right)$ space, and $O\left(\log \log n\right)$ query time, improving thus the near-cubic preprocessing bound by Roditty and Segal [SODA 2007].

2012 ACM Subject Classification Theory of computation → Computational geometry, Theory of computation → Shortest paths

Keywords and phrases shortest paths, distance oracles, unit-disk graphs, planar graphs

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.24

1 Introduction

In this paper, we study shortest-path problems in *weighted unit-disk graphs*, i.e., intersection graphs of unit disks. More concretely, in such a graph, vertices correspond to a set S of planar points (specifically, the centers of the disks), and there is an edge between every two points of S at Euclidean distance at most one (of weight equal to that distance). These graphs have been widely used in many applications, such as modelling ad-hoc communication networks.

We are interested in various basic problems about shortest paths in such a weighted unit-disk graph G, notably:

- designing algorithms for computing a $(1+\varepsilon)$-approximation of various parameters of G, such as the *diameter* (i.e., $\max_{s,t \in S} d_G[s,t]$), the *radius* (i.e., $\min_{s \in S} \max_{t \in S} d_G[s,t]$), the *bichromatic closest pair distance* of two subsets $A, B \subset S$ (i.e., $\min_{a \in A,\, b \in B} d_G[a,b]$), et cetera; and,
- designing *approximate distance oracles*, i.e., data structures that support the following query: given any $s, t \in S$, quickly compute a $(1+\varepsilon)$-approximation of the s-to-t shortest-path distance in G, $d_G[s,t]$.

Besides practical motivation from wireless networks, this collection of problems is interesting from the theoretical perspective as well since it connects computational geometry with graph data structures – indeed, our new algorithms will draw on ideas from both areas.

Planar graph techniques. There has already been an extensive body of work devoted to distance oracles and shortest-path-related problems both for general and for planar graphs, the latter of which are of particular relevance to us. For example, Thorup [25] gave $(1+\varepsilon)$-approximate distance oracles for weighted, undirected planar graphs with $O(n \operatorname{polylog} n)$ preprocessing time and space, and $O(1)$ query time, for any constant $\varepsilon > 0$, while subsequent work [19, 18, 16, 10] gave improvements and examined the dependency of the hidden factors on ε. Weimann and Yuster [27] presented a $(1+\varepsilon)$-approximation algorithm for the diameter for weighted, undirected planar graphs, running in $O\left(n \log^4 n\right)$ time, for any constant $\varepsilon > 0$, improved later to $O\left(n \log^2 n\right)$ by Chan and Skrepetos [10], who also reduced the ε-dependency from exponential to polynomial (there has also been exciting recent breakthrough on *exact* algorithms for diameter and distance oracles in planar graphs, by Cabello [5] and subsequent researchers [15, 11, 14]).

All the above approximation results for planar graphs rely heavily on the concept of *shortest-path separator*: a set of shortest paths with common root, such that the removal of their vertices decomposes the graph into at least two disjoint subgraphs. Unfortunately, such separators do not seem directly applicable to unit-disk graphs, and not only because the latter may be dense. Indeed, by grid rounding, we can construct a sparse weighted graph $\widehat{G}$, such that it (i) approximately preserves distances in the original unit-disk graph G (e.g., see the proof of Lemma 2), and (ii) is "nearly planar", in the sense that each edge intersects at most a constant number of other edges. However, even for such a graph, it is not clear how to define a shortest-path separator that divides it cleanly into an inside and an outside because edges may "cross" over the separator. At least one prior paper [28] worked on extending shortest-path separators to unit-disk graphs, but the construction was complicated and achieved only constant approximation factors.

Gao and Zhang's WSPD technique. In a seminal paper, Gao and Zhang [13] obtained the first nontrivial set of results on shortest-path problems in weighted unit-disk graphs, by adapting a familiar technique in computational geometry – namely, the *well-separated pair decomposition* (WSPD), introduced by Callahan and Kosaraju [7] for addressing proximity problems in the Euclidean (or L_p) metric and has since found countless applications. Gao and Zhang proposed a new variant of WSPDs for the weighted unit-disk graph metric and showed that any n-point set in two dimensions has a WSPD of near-linear ($O(n \log n)$) size under the new definition. Consequently, they obtained a $(1+\varepsilon)$-approximate distance oracle with $O(n \log n)$ size and $O(1)$ query time, for any constant $\varepsilon > 0$. Unfortunately, its preprocessing time, $O\left(n^{3/2}\sqrt{\log n}\right)$, is quite high and becomes the bottleneck when the technique is applied to offline problems such as computing the diameter.

However, the issue is not constructing the WSPD itself, which can be done in near-linear time, but computing the shortest-path distances of a near-linear number of vertex pairs in the "nearly planar" graph $\widehat{G}$ mentioned above, which takes almost $n^{3/2}$ time, by adapting a known exact distance oracle for planar graphs [1] (noting that $\widehat{G}$ has balanced separators [22, 12]). Cabello [4] has given an improved algorithm for computing multiple distances in planar graphs, and if it could be adapted here, the running time would be reduced to around $n^{4/3}$. However, near-linear time still seems out of reach with current techniques.

Gao and Zhang [13] observed that the preprocessing time can be made near-linear when the approximation factor is a certain constant (about 2.42), but this improvement does not apply to $1+\varepsilon$ approximation factor and has no new implication to the diameter problem (for which a near-2-approximation is easy by running a single-source shortest paths algorithm).

New results. In Section 2, we give the first near-linear-time algorithm to compute a $(1+\varepsilon)$-approximation of the diameter of a weighted unit-disk graph, running in $O\left(n \log^2 n\right)$ time, for any constant $\varepsilon > 0$ (the dependencies of the hidden factors on ε are polynomial). A similar result holds for $(1+\varepsilon)$-approximate distance oracles: we obtain $O\left(n \log^3 n\right)$ preprocessing time, $O\left(n \log n\right)$ space, and $O(1)$ query time. We thus answer one of the main questions left open in Gao and Zhang's paper, while also apply our techniques to related problems.

Our approach is conceptually simple: we just go back to known shortest-path separator techniques for planar graphs [25, 18]!

But how do we get around the issue that unit-disk graphs do not have nice path separators? We first find a *spanner* subgraph H that is planar and has constant approximation/stretch factor (fortunately, such spanners are known to exist in unit-disk graphs [21] and they were also used by Gao and Zhang [13]) and then then apply divide-and-conquer over the shortest-path separator decomposition tree for H instead of G.

Although the above plan may sound obvious in hindsight, the details are tricky to get right. For example, how could the use of a spanner with $O(1)$ approximation factor eventually lead to $1+\varepsilon$ approximation factor? The known divide-and-conquer approaches for planar graphs select a small number of vertices, called *portals*, along each separator and compute distances from each with a Single-Source Shortest Paths algorithm; that works well because a shortest path in a planar graph crosses a separator only at vertices. In our case, however, we need to use the original (non-planar, unit-disk) graph G when computing distances from portals, but therein a shortest path could "cross" the separator over an edge. We show that we can nevertheless re-route such a path to pass through a separator vertex without increasing the length by much, by using the fact that H is a $O\left(1\right)$-spanner.

Application to all-pairs bounded-leg shortest paths. In the last part of the paper, as a further application, we employ our new distance oracle, along with additional ideas, to solve the $(1+\varepsilon)$-approximate *All-Pairs Bounded-Leg Shortest Paths* (apBLSP) problem. Given a set S of n planar points, we define $G_{\leq L}$ to be the subgraph of the complete Euclidean graph of S that contains only edges of weight at most L. Then, we want to preprocess S, such that given two points $s, t \in S$ and any positive number L, we can quickly compute a $(1+\varepsilon)$-approximation of the s-to-t shortest path in $G_{\leq L}$ (i.e., the shortest path under the restriction that each leg of the trip has length bounded by L) or its length. To see the connection of apBLSP with the earlier problems, note that, for each fixed L, $G_{\leq L}$ is a weighted unit-disk graph, after rescaling the radii. One important difference, however, is that L is not fixed in apBLSP, and we want to answer queries for any of the $\binom{n}{2}$ combinatorially different L's.

Bose et al. [2] introduced that problem in 2003 and gave a data structure for it with $O\left(n^5\right)$ preprocessing time, $O\left(n^2 \log n\right)$ space, and $O\left(\log n\right)$ query time, for any constant $\varepsilon > 0$, while Roditty and Segal [24] improved the preprocessing time to roughly $O\left(n^3\right)$ and the query time to $O\left(\log \log n\right)$.

In Section 3, we apply our $(1+\varepsilon)$-approximate distance oracle for weighted unit-disk graphs, along with additional new ideas, to obtain the first method to break the cubic preprocessing barrier: we can obtain roughly $O\left(n^{8/3}\right)$ preprocessing time, while keeping

$O\left(n^2 \log n\right)$ space and $O\left(\log \log n\right)$ query time. With fast matrix multiplication, we can further reduce the preprocessing time to $O\left(n^{2.579}\right)$, assuming a polynomial bound on the *spread*, i.e., the ratio of the maximum to the minimum Euclidean distance over all pairs of points in V.

2 Approximate diameter and distance oracles

Let S be a set of planar points whose weighted unit-disk graph G has diameter Δ. A key subproblem in both (i) computing a $(1+\varepsilon)$-approximation of the diameter of G and (ii) building a $(1+\varepsilon)$-approximate distance oracle for it is the construction of a distance oracle with *additive stretch* $O\left(\varepsilon\Delta\right)$: a data structure, such that, given any $s, t \in S$, we can quickly compute a value $\widetilde{d}$ with $d_G[s,t] \leq \widetilde{d} \leq d_G[s,t] + O\left(\varepsilon\Delta\right)$. We describe our solution for that subproblem in Section 2.2, after giving two preliminary ingredients in Section 2.1, and then show, in Section 2.3, how to employ it, along with existing techniques, to address the two original problems.

2.1 Preliminaries

The first ingredient we need is the existence of a planar spanner with constant stretch factor in any weighted unit-disk graph.

▶ **Lemma 1** (Planar spanner). *Given a set S of n planar points, we can find, in $O\left(n \log n\right)$ time, a planar spanning subgraph H of its weighted unit-disk graph G, such that, for every $s, t \in S$, $d_G[s,t] \leq d_H[s,t] \leq c d_G[s,t]$, where c is some constant.*

Li, Calinescu, and Wan [21] proved the above lemma with $c = 2.42$ by simply building the Delaunay triangulation of the given points and discarding edges of weight more than one; however, the analysis of the stretch factor c is nontrivial.

The second ingredient is an efficient algorithm for the Single-Source Shortest Paths (SSSP) problem in weighted unit-disk graphs, where the currently best exact result, due to Cabello and Jejčič [6], requires $O\left(n \log^{12+o(1)} n\right)$ time and employs complicated dynamic data structures for additively-weighted Voronoi diagrams [8, 17]. For our purposes though, it suffices to consider the $(1+O\left(\varepsilon\right))$-approximate version of the problem instead, i.e., given a set of points S and a source $s \in S$, compute, for each $t \in S$, a path of length $\widetilde{d}[s,t]$, such that $d_G[s,t] \leq \widetilde{d}[s,t] \leq (1+O\left(\varepsilon\right)) d_G[s,t]$, where G is the weighted unit-disk graph of S. Our algorithm first finds a sparse graph $\widehat{G}$ that $(1+O\left(\varepsilon\right))$-approximately preserves distances in G (i.e., for any $s, t \in S$, there are vertices p_s, p_t of $\widehat{G}$, such that $\delta_G[s,t] \leq \delta_{\widehat{G}}[c_s, c_t] \leq (1+O\left(\varepsilon\right)) d_G[s,t]$) and then runs Dijkstra's algorithm therein; sparsification in weighted unit-disk graphs has been used before (e.g., see [13, Section 4.2]).

▶ **Lemma 2** (Approximate SSSP). *Given a set S of n planar points, we can solve the $(1+O\left(\varepsilon\right))$-approximate SSSP problem in its weighted unit-disk graph G in $O\left((1/\varepsilon)^2 n \log n\right)$ time.*

Proof. First, we build a uniform grid of side length ε and construct a sparse weighted graph $\widehat{G}$ by placing a vertex at each non-empty grid cell and an edge between every two such cells c and c' iff there exist points $p \in c$ and $p' \in c'$ with $\|pp'\| \leq 1$; the weight of that edge is equal to the maximum Euclidean distance of c and c'. Each grid cell has at most $O\left((1/\varepsilon)^2\right)$ neighbors, so $\widehat{G}$ has at most $O\left((1/\varepsilon)^2 n\right)$ edges and can be constructed in $O\left((1/\varepsilon)^2 n \log n\right)$ time, by using a Euclidean bichromatic closest pair algorithm [23] over $O\left((1/\varepsilon)^2 n\right)$ pairs of grid cells.

Let s and t be two points of S; if $\|st\| \leq 1$, we can trivially return $\|st\|$. Else, let $p_0 p_1 \cdots p_\ell$, with $p_0 = s$ and $p_\ell = t$, be the shortest path in G from s to t. Two consecutive edges therein have lengths whose sum is at least one because otherwise we could take a short-cut and obtain a shorter path; thus, $d_G[s,t] \geq \lfloor \ell/2 \rfloor$. We construct a path $c_0 c_1 \cdots c_\ell$ in $\widehat{G}$, where each c_i is the cell that contains p_i. Since, for each c_i, c_{i+1}, $\|p_i p_{i+1}\| \leq d_{\widehat{G}}[c_i, c_{i+1}] \leq \|p_i p_{i+1}\| + O(\varepsilon)$, it follows that $d_G[s,t] \leq d_{\widehat{G}}[c_0, c_\ell] \leq d_G[s,t] + O(\varepsilon\ell) \leq (1 + O(\varepsilon)) d_G[s,t]$.

Thus, given a source $s \in S$, we can invoke Dijkstra's algorithm in $\widehat{G}$ to compute, for each $t \in S$, a value $d_{\widehat{G}}[c_s, c_t]$, such that $\delta_G[s,t] \leq \delta_{\widehat{G}}[c_s, c_t] \leq (1 + O(\varepsilon)) d_G[s,t]$, where c_s and c_t are the grid cells that contain s and t, respectively. We can easily modify our algorithm to also find, for each $t \in S$, an s-to-t path in G of length $\delta_{\widehat{G}}[c_s, c_t]$, by appending s and t at the ends of the s-to-t shortest path in $\widehat{G}$ and replacing each c_i and c_{i+1} with the bichromatic closest pair of $((S \cap c_i), (S \cap c_{i+1}))$ in G (which has been found while constructing $\widehat{G}$). ◀

2.2 Distance oracles with $O(\varepsilon\Delta)$ additive stretch

We describe now a distance oracle with additive-stretch for an arbitrary weighted graph $G = (V, E)$ of n vertices and of diameter Δ that has the following properties, which are the only ones needed from weighted unit-disk graphs.

- **(I)** There exists a planar *c-spanner* H of G, for some constant c.
- **(II)** For any induced subgraph of G with n' vertices, the $(1+\varepsilon)$-approximate SSSP problem can be solved in $T(n')$ time, for some function $T(\cdot)$, such that $T(n')/n'$ is nondecreasing.
- **(III)** Every edge weight in G is at most $\varepsilon\Delta$.

If G is a weighted unit-disk graph, Lemmas 1 and 2 imply (I) and (II), respectively, where $c = 2.42$ and $T(n') = O\left((1/\varepsilon)^2 n' \log n'\right)$, and (III) holds as long as $\Delta \geq 1/\varepsilon$.

Shortest-path separators in H. Although G may not necessarily have nice shortest-path separators, we know that H does, by planarity. Thus, we apply a known shortest-path separator decomposition therein, namely the version of Kawarabayashi, Sommer, and Thorup [18, Section 3.1], paraphrased for our purposes. Specifically, we can compute in $O(n \log n)$ time a *decomposition tree* $\mathcal{T}$ with the following properties.

- $\mathcal{T}$ has $O(1)$ degree and $O(\log n)$ height.
- Each node μ of $\mathcal{T}$ is associated with a subset $V^{(\mu)} \subseteq V$. The subsets $V^{(\nu)}$ over all children ν of μ are disjoint and contained in $V^{(\mu)}$. If μ is the root, $V^{(\mu)} = V$; if μ is a leaf, $V^{(\mu)}$ has $O(1)$ size.
- Each non-leaf node μ of $\mathcal{T}$ is associated with a set of $O(1)$ paths, called *separator paths*, which are classified as "internal" and "external". The internal separator paths cover precisely all vertices of $V^{(\mu)} - \bigcup_{\text{child } \nu \text{ of } \mu} V^{(\nu)}$, while the external are outside of $V^{(\mu)}$.
- For each child ν of a non-leaf node μ, every neighbor of the vertices of $V^{(\nu)}$ in H is either in $V^{(\nu)}$ or in one of the (internal or external) separator paths at μ.
- Each separator path is a shortest path in H and, in particular, has length at most the diameter $\Delta(H)$ of H (which is at most $c\Delta$).

Our data structure. To construct an additive oracle with $O(\varepsilon\Delta)$ stretch for G, we construct the above decomposition tree $\mathcal{T}$ and augment it with extra information, as follows. Let μ be an internal node of $\mathcal{T}$ and σ one of its internal separator paths; since σ has length at

most $\Delta(H) \leq c\Delta$, we can select, with a linear walk, a set of $O(1/\varepsilon)$ vertices thereon, called *portals*, such that each consecutive pair of them is at distance at most $\varepsilon\Delta$ on it.

Let $P^{(\mu)}$ denote the set of all portals over all internal separator paths at a non-leaf node μ of $\mathcal{T}$. For each such node and for each $p \in P^{(\mu)}$ and $v \in V^{(\mu)}$, we invoke $O(1/\varepsilon)$ times the SSSP algorithm from Property (II) to compute a $(1+\varepsilon)$-approximation, $\widetilde{d}_\mu[p,v]$, of the shortest path distance from p to v in the subgraph of G induced by $V^{(\mu)}$. Then, for each leaf μ, we just find and store all pairwise distances in the subgraph of G that is induced by $V^{(\mu)}$. Overall, our oracle requires $O((1/\varepsilon)T(n) \cdot \log n)$ preprocessing time and $O((1/\varepsilon)n \cdot \log n)$ space.

Query algorithm. Given two vertices $s, t \in V$, we first identify all $O(\log n)$ nodes μ in $\mathcal{T}$, such that both $s \in V^{(\mu)}$ and $t \in V^{(\mu)}$ (by trivially starting from the root and going down the tree along a path). For each such non-leaf node μ, we compute, in $O(1/\varepsilon)$ time, a value $\widetilde{\delta}_\mu[s,t] = \min_{p \in P^{(\mu)}} \{\widetilde{d}_\mu[s,p] + \widetilde{d}_\mu[p,t]\}$. If μ is a leaf, $\widetilde{d}_\mu[s,t]$ is the exact shortest path distance in the subgraph of G induced by $V^{(\mu)}$ (which we have already computed). Finally we return the minimum, $\widetilde{\delta}[s,t]$, over all $\widetilde{\delta}_\mu[s,t]$. The total query time is $O((1/\varepsilon)\log n)$.

Stretch analysis. We want to prove that for any $s, t \in V$, the value, $\widetilde{d}[s,t]$, that our oracle returns is such that $d_G[s,t] \leq \widetilde{d}[s,t] \leq d_G[s,t] + O(\varepsilon\Delta)$. The left side of the inequality clearly holds because $\widetilde{d}[s,t]$ corresponds to the length of an s-to-t path in a subgraph of G. To prove the right side, let π be the shortest s-to-t path in G, and let μ be the lowest node in $\mathcal{T}$, such that all vertices of π lie in $V^{(\mu)}$; we assume that μ is a non-leaf node (otherwise we have already computed $d_G[s,t]$ exactly).

Although π is a path in the (not necessarily planar) graph G, not H, we show that it is possible to re-route it to pass through a vertex on a separator path of μ without increasing its length by much.

▶ **Claim 3** (Detour through a separator vertex)**.** *There exists an s-to-t path π' in G that (i) passes through some vertex w on a separator path of μ, (ii) uses only vertices of $V^{(\mu)}$ (except maybe for w itself) and (iii) has length at most $d_G[s,t] + 2c\varepsilon\Delta$.*

Proof. We assume that none of the vertices on π lie on a separator path of μ because otherwise we can just set $\pi' = \pi$. Let ν be the child of μ with $s \in V^{(\nu)}$, let u be the last vertex on π that lies in $V^{(\nu)}$ (note that $u \neq t$, by definition of μ), and let v be the next vertex after u thereon. By (I), there is a path $\pi_{u,v}$ from u to v in H of length at most $c \cdot$ (the weight of uv), which is at most $c\varepsilon\Delta$ by (III). Let w be the first vertex on $\pi_{u,v}$ that lies outside of $V^{(\nu)}$ (which exists since v is outside of $V^{(\nu)}$); then, from the fourth property of $\mathcal{T}$, we know that w must be on an (internal or external) separator path σ of μ. Thus, we set π' to be the path that goes from s to u along π, then from u to w along $\pi_{u,v}$ (which uses only vertices in $V^{(\nu)}$ as intermediates), then back from w to u along $\pi_{u,v}$, and finally from u to t along π. See Figure 1(a) (where σ is internal) and 1(b) (where σ is external). ◀

Next, we note how to further re-route π to pass through a portal.

▶ **Claim 4** (Detour through a portal)**.** *There exists another s-to-t path π'' in G that (i) passes through a portal p on a separator path σ' of μ', where μ' is some ancestor of μ, (ii) uses only vertices of $V^{(\mu')}$, and (iii) has length at most $d_G[s,t] + (2c+2)\varepsilon\Delta$.*

Proof. Let w be as in Claim 3, and let μ' be the lowest ancestor of μ, such that $w \in V^{(\mu')}$ (notice that if $w \in V^{(\mu)}$, $\sigma = \sigma'$). Then w must be on an internal separator path σ' in μ'

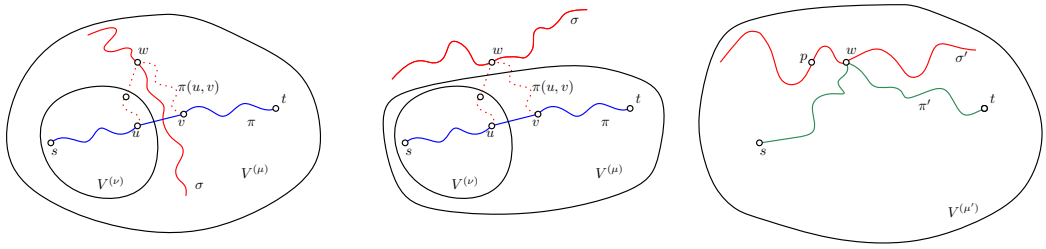


Figure 1 Detour through a vertex of a separator path σ in Claim 3, where σ may be internal, as in (a), or external, as in (b); detour through a portal in Claim 4 in (c).

(whose existence is guaranteed by the third property of $\mathcal{T}$). Let p be the portal on σ' that is closest to w, so the p-to-w distance on σ' is at most $O(\varepsilon\Delta)$. We set π'' to be the path the goes from s to w along π', then from w to p along σ' and back from p to w, and, finally, from w to u along π'. See Figure 1(c). ◀

Let μ' be as in Claim 4. It follows that $\widetilde{\delta}[s,t] \leq \widetilde{\delta}_{\mu'}[s,t] \leq \widetilde{d}_{\mu'}[s,p] + \widetilde{d}_{\mu'}[p,t] \leq d_G[s,t] + O(\varepsilon\Delta)$.

▶ **Theorem 5** (General Additive-Stretch Distance Oracle). *Given a weighted graph of n vertices and of diameter Δ that satisfies Properties (I)–(III), we can construct for it, in $O((1/\varepsilon)T(n)\log n)$ time, a distance oracle of $O(\varepsilon\Delta)$ additive stretch, $O((1/\varepsilon)n\log n)$ space, and $O((1/\varepsilon)\log n)$ query time.*

As we saw earlier, weighted unit-disk graphs satisfy Properties (I)-(III), thus we have the following theorem.

▶ **Corollary 6** (Additive-Stretch Distance Oracle in Unit-Disk Graphs). *Given a set S of n planar points, such that the weighted unit-disk graph of S has diameter $\Delta \geq 1/\varepsilon$, we can construct for the latter, a distance oracle of $O(\varepsilon\Delta)$ additive stretch, $O\left((1/\varepsilon)^3 n\log^2 n\right)$ preprocessing time, $O((1/\varepsilon)n\log n)$ space, and $O((1/\varepsilon)\log n)$ query time.*

2.3 Applications

We now describe how to employ Corollary 6 to compute a $(1+\varepsilon)$-approximation of the diameter of a unit-disk graph and how to build a $(1+\varepsilon)$-approximate distance oracle for it.

Approximate diameter. To approximate the diameter of a weighted unit-disk graph, we use the following lemma, implied by Gao and Zhang's WSPD-based technique [13, Corollary 5.2].

▶ **Lemma 7** (Via Well-Separated Pair Decomposition). *Given a set S of n planar points, we can find a set of $O\left((1/\varepsilon)^4 n\log n\right)$ pairs of them in $O\left((1/\varepsilon)^4 n\log n\right)$ time, such that the shortest-path distance between any two vertices in the weighted unit-disk graph of S can be $(1+\varepsilon)$-approximated by the shortest-path distance between one of these pairs, which can be found in $O(1)$ time.*

First we compute in $O(n\log n)$ time [23] the Euclidean diameter, Δ_0, of S; if $\Delta_0 \geq 1/\varepsilon$, then $\Delta \geq 1/\varepsilon$, and, to compute a $(1+\varepsilon)$-approximation of Δ, we can query the oracle of Corollary 6 of $O(\varepsilon\Delta)$ additive stretch with all $O\left((1/\varepsilon)^4 n\log n\right)$ pairs of Lemma 7 and return

the maximum; thus the approximation factor is $1 + O(\varepsilon)$. The total time required for this case is $O\left(((1/\varepsilon)^4 n \log n \cdot (1/\varepsilon) \log n\right)$.

If $1 < \Delta_0 < 1/\varepsilon$, the problem is more straightforward because we can construct the sparsified graph $\widehat{G}$ from the proof of Lemma 2, which preserves distances approximately, and then run a standard All-Pairs Shortest Paths (APSP) algorithm therein. Since $\widehat{G}$ has $\widehat{n} = O\left((\Delta_0/\varepsilon)^2\right) = O\left((1/\varepsilon)^4\right)$ vertices and $\widehat{m} = O\left((1/\varepsilon)^2 \widehat{n}\right) = O\left((1/\varepsilon)^6\right)$ edges, we need $O\left(\widehat{n}^2 \log \widehat{n} + \widehat{m}\widehat{n}\right) = O\left((1/\varepsilon)^{10}\right)$ time for this case. Finally, if $\Delta_0 < 1$, the unit-disk graph is a complete Euclidean graph, so we just return Δ_0.

▶ **Theorem 8** (Approximate Diameter). *Given a set S of n planar points, we can compute, in $O\left((1/\varepsilon)^5 n \log^2 n + (1/\varepsilon)^{10}\right)$ time, a $(1+\varepsilon)$-approximation of the diameter of the weighted unit-disk graph of S.*

▶ Remark. Employing a WSPD is not essential here, as we could combine our techniques with those of Weimann and Yuster for planar graphs [27], increasing, though, the ε-dependency to $2^{O(1/\varepsilon)}$.

Approximate distance oracles. To build a distance oracle of $(1+\varepsilon)$-approximation factor for a weighted unit-disk graph, we employ the oracle of Corollary 6 of $O(\varepsilon\Delta)$ additive stretch as a building block in a known technique called *sparse neighborhood covers*. We use sparse neighborhood covers result of Busch et al. [3] for planar graphs, whose construction time bound was given by Kawarabayashi et al. [18].

▶ **Lemma 9** (Sparse neighborhood cover). *Given a weighted planar graph H of n vertices and a value r, we can construct, in $O(n \log n)$ time, a collection of subsets V_i of V, such that (i) the diameter of the subgraph of H induced by each V is $O(r)$, (ii) every vertex resides in $O(1)$ subsets, and (iii) for every vertex v, the set of all vertices at distance at most r from v in H is contained in at least one of the V_i's.*

Let G be the weighted unit-disk graph of a set S of n planar points, and let H be an $O(1)$-planar spanner of G. Every shortest path distance in G is upper bounded by n, so we first apply the above lemma to H for each value of $r \in \{2^0, 2^1, \ldots, 2^{\log n}\}$, thus obtaining collections of subsets $V_i^{(r)}$, and then build the distance oracle of Corollary 6 for the weighted unit-disk graph of each $V_i^{(r)}$. The total preprocessing time and space over all $O(\log n)$ choices of r is $O\left(\log n \cdot (1/\varepsilon)^3 n \log^2 n\right)$ and $O(\log n \cdot (1/\varepsilon) n \log n)$, respectively. Given $s, t \in S$, we consider each r and each subset $V_i^{(r)}$ that contains both s and t, query the oracle for $V_i^{(r)}$, and return the minimum. The total query time over all $O(\log n)$ choices of r and $O(1)$ choices of $V_i^{(r)}$ (Lemma 9(ii)) is $O(\log n \cdot (1/\varepsilon) \log n)$.

If $d_G[s,t] \geq 1/\varepsilon$, let $r \geq c/\varepsilon$ be such that $d_G[s,t] \in (r/2c, r/c]$. Then, each vertex on the shortest path from s to t in G is at distance at most $c d_G[s,t] \leq r$ from s in H, so it is contained in a common subset $V_i(r)$, and we approximate $d_G[s,t]$ with an additive error of $O(\varepsilon r) = O(\varepsilon d_G[s,t])$, obtaining thus $1 + O(\varepsilon)$ approximation factor.

If $1 < d_G[s,t] < 1/\varepsilon$, we simply build the sparsified graph $\widehat{G}$ from the proof of Lemma 2, which preserves distances approximately, and, from every vertex, we pre-compute the distances to all grid cells at Euclidean distance at most $1/\varepsilon$, by running Dijkstra's algorithm on a subgraph of $\widehat{G}$ with $n' = O\left(1/\varepsilon\right)^4)$ vertices and $O\left((1/\varepsilon)^2 n'\right) = O\left((1/\varepsilon)^6\right)$ edges, in $O\left((1/\varepsilon)^6 \log(1/\varepsilon)\right)$ time. The total preprocessing time and space over all sources is $O\left((1/\varepsilon)^6 n \log(1/\varepsilon)\right)$ and $O\left((1/\varepsilon)^4 n\right)$, respectively. Finally, if $\delta_G[s,t] \leq 1$, the shortest-path distance of s and t is their Euclidean distance. We do not know a priori which of the cases we are in, so we try all of them and return the minimum distance found.

▶ **Theorem 10** (Approximate Distance Oracle). *Given a set S of n planar points, we can construct a $(1+\varepsilon)$-approximate distance oracle for its weighted unit-disk graph with $O\left((1/\varepsilon)^3 n\log^3 n + (1/\varepsilon)^6 n\log(1/\varepsilon)\right)$ preprocessing time, $O\left((1/\varepsilon)n\log^2 n + (1/\varepsilon)^4 n\right)$ space and $O\left((1/\varepsilon)\log^2 n\right)$ query time.*

To reduce the query time, we can combine the above method with Gao and Zhang's WSPD-based oracle [13, Section 5.1], which requires $O(1)$ query time and $O((1/\varepsilon)n\log n)$ space. Its construction time is dominated by finding $(1+\varepsilon)$-approximate shortest-path distances for $O\left((1/\varepsilon)^4 n\log n\right)$ pairs, however, we can compute these distances by querying our oracle of Theorem 10 in $O\left((1/\varepsilon)^4 n\log n\cdot(1/\varepsilon)\log^2 n\right)$ total time.

▶ **Corollary 11** (Approximate Distance Oracle with $O(1)$ Query Time). *Given a set S of n planar points, we can construct a $(1+\varepsilon)$-approximate distance oracle for its weighted unit-disk graph of $O\left((1/\varepsilon)^5 n\log^3 n + (1/\varepsilon)^6 n\log(1/\varepsilon)\right)$ preprocessing time, $O\left((1/\varepsilon)^4 n\log n\right)$ space, and $O(1)$ query time.*

Similarly, we can use the distance oracle of Theorem 10 to improve Gao and Zhang's results for other distance-related problems on weighted unit-disk graphs:

▶ **Corollary 12** (Approximate Radius and Bichromatic Closest Pair). *Given a set S of n planar points, we can compute, in $O\left((1/\varepsilon)^5 n\log^3 n + (1/\varepsilon)^6 n\log(1/\varepsilon)\right)$ time, a $(1+\varepsilon)$-approximation of the radius of the weighted unit-disk graph of S or of the bichromatic closest pair distance of two given subsets $A, B \subseteq S$ therein.*

▶ Remark.

- For the sake of simplicity, we did not optimize the $\mathrm{poly}(1/\varepsilon, \log n)$ factors.
- Our distance oracle in Theorem 10 can be easily modified to report an approximate shortest path, not just its distance, in additional time proportional to the number of edges in the path: every time we find approximate shortest distances in a subgraph from a portal, we also store its approximate shortest path tree.
- The same approach gives $(1+O(\varepsilon))$-approximation results for *unweighted* unit-disk graphs, assuming that the diameter and the distances of the query vertices exceed $\Omega(1/\varepsilon)$. Specifically, Lemma 7 can be modified for the unweighted case, but the error now has an extra additive term of $4+O(\varepsilon)$ [13, Lemma 6.2], which can be ignored under our assumption. Also, we need to replace the SSSP algorithm of Lemma 2 with the $O(n\log n)$-time exact SSSP algorithm by Cabello and Jejčič [6] or by Chan and Skrepetos [9].

3 Approximate apBLSP

In this section, we study the $(1+\varepsilon)$-approximate apBSLP problem. Given a set S of n planar points, let G be its complete weighted Euclidean graph, let $w_1, w_2, \ldots, w_N$, where $N = \binom{n}{2}$, be the weights of the edges of G in non-decreasing order, and let G^i be the subgraph of G that contains only the edges of weight at most w_i. We can assume that $w_1 \geq 1$; else, we can impose that assumption by simply translating and rescaling S in linear time. We want to preprocess S into a data structure, such that we can quickly answer $(1+\varepsilon)$-approximate *bounded-leg distance* queries, i.e., given $s, t \in S$ and a positive number L, compute a $(1+\varepsilon)$-approximation of $d_{G^i}[s,t]$, where i is the largest integer with $w_i \leq L$. First, we briefly review the previous methods of Bose et al. [2] and of Roditty and Segal [24], in Section 3.1, and then describe our own approach, in Section 3.2.

3.1 Previous methods

Let $s, t \in S$, and let $c(s,t)$ be the minimum index, such that s and t are connected in $G^{c(s,t)}$. Then, since each G^i is a subgraph of G^{i+1}, we have that $d_G[s,t] \leq d_{G^{N-1}}[s,t] \leq \cdots \leq d_{G^{c(s,t)}}[s,t]$. Moreover, the s-to-t shortest path in any G^i with $i > c(s,t)$ must have an edge of weight at least $w_{c(s,t)}$, so $d_G[s,t] \geq w_{c(s,t)}$; any shortest path has at most $n-1$ edges, thus $d_{G^{c(s,t)}}[s,t] \leq (n-1)d_G[s,t]$. Therefore, as Roditty and Segal [24, Section 2] noticed, we can compute and store, for each $s, t \in S$, a $(1+\varepsilon)$-approximation of the s-to-t shortest path distance in only $O\left(\log_{1+\varepsilon} n\right)$ graphs, such that a bounded-leg distance query can be answered with a binary search in $O\left(\log\log_{1+\varepsilon} n\right)$ time.

Specifically, for every $s, t \in S$ and $j \in \{0, 1, \ldots, \lceil \log_{1+\varepsilon} n \rceil\}$, let $I^j(s,t)$ be the set of indices of the graphs G^i, such that $(1+\varepsilon)^j \delta_G[s,t] \leq \delta_{G^i}[s,t] \leq (1+\varepsilon)^{j+1}\delta_G[s,t]$. If $I^j(s,t) \neq \emptyset$, we create two values $m^j(s,t)$ and $\ell^j(s,t)$, where the former is any index therein and the latter is equal to $w_{m^j(s,t)}$; else, $m^j(s,t)$ and $\ell^j(s,t)$ are undefined. The total space required over all pairs of S is $O\left(n^2 \log_{1+\varepsilon} n\right)$. Then, given a positive number L, we can find the largest i among the $m^j(s,t)$'s, such that $w_i \leq L$, with a binary search over the $\ell^j(s,t)$'s, in $O\left(\log\log_{1+\varepsilon} n\right)$ time, and return a $(1+\varepsilon)$-approximation of the s-to-t shortest path distance in G^i.

To compute a possible index for $m^j(s,t)$, for every $s, t \in S$ and $j \in \{0, 1, \ldots, \lceil \log_{1+\varepsilon} n \rceil\}$, Roditty and Segal performed $O\left(n^2 \log_{1+\varepsilon} n\right)$ independent binary searches, each making $O(\log n)$ $(1+\varepsilon)$-approximate bounded-leg distance queries (i.e., a query to find a $(1+\varepsilon)$-approximation of the s-to-t shortest path distance in some graph G^i). Instead, we group the queries for all s, t, j into $O\left(\log n \cdot \log_{1+\varepsilon} n\right)$ rounds of n^2 offline queries each, where "offline" means that the queries in every round are given in advance.

▶ **Lemma 13** (Framework for Approximate apBLSP). *Given a set S of n planar points, we can construct a data structure for the $(1+\varepsilon)$-approximate apBLSP problem of $O\left((1/\varepsilon)n^2 \log n\right)$ space, $O(\log\log n + \log(1/\varepsilon))$ query, and $O\left(T_{\text{offline}}(n, n^2, 1+\varepsilon) \cdot (1/\varepsilon)\log^2 n\right)$ preprocessing time, where $T_{\text{offline}}(n', q', 1+\varepsilon')$ denotes the total time for answering q offline $(1+\varepsilon')$-approximate bounded-leg distance queries for an n'-point set.*

To address each round, Roddity and Segal's method would imply constructing in near-linear time a sparse $(1+\varepsilon)$-spanner of every graph G^i and then running Dijkstra's algorithm therein to answer each query; thus a near-cubic bound would be obtained for $T_{\text{offline}}(n, n^2, 1+\varepsilon)$. Instead, we show that by employing our $(1+\varepsilon)$-approximate distance oracle of Corollary 11 for weighted unit-disk graphs as a subroutine, we can obtain a truly subcubic bound on $T_{\text{offline}}(n, n^2, 1+\varepsilon)$, as we next describe.

3.2 Improved method

We view the problem of answering, for each $s, t \in S$ and $j \in \{0, 1, \ldots, \lceil \log_{1+\varepsilon} n \rceil\}$, n^2 approximate offline bounded-leg distance queries as the problem of constructing and querying the following offline *semi-dynamic* (actually insertion-only) distance oracle.

▶ **Subproblem 1** (Semi-Dynamic Approximate Distance Oracles). *Given an arbitrary graph of n vertices with edge weights in $[1, \infty)$, we want to perform an offline sequence of q operations, each of which is either an edge insertion, or a query to compute a $(1+\varepsilon)$-approximation of the shortest-path distance between two vertices. Let $T_{\text{dyn}}(n, q, 1+\varepsilon)$ be the complexity of this problem.*

We could reduce our problem to Subproblem 1 by naively inserting the $O\left(n^2\right)$ edges of G in increasing order of weight to an initially empty graph and mix that sequence of

insertions with the given sequence of bounded-leg distance queries. Hence, we would have that $T_{\text{offline}}(n, n^2, 1+\varepsilon) = O\left(T_{\text{dyn}}(n, n^2, 1+\varepsilon)\right)$.

Instead, we propose a better reduction that employs a simple periodic rebuilding trick. First, we divide the sequence of the q edge insertions and queries into $O(q/r)$ *phases* of at most r operations each, where r is a parameter to be set later. At the beginning of each phase, the current graph is a weighted unit-disk graph (after rescaling), so we can build the $(1+\varepsilon)$-approximate distance oracle of Corollary 11 in $O\left((1/\varepsilon)^5 n \log^3 n\right)$ time. Then, in $O\left(r^2\right)$ total time, we query that oracle to approximate the shortest-path distances between all pairs of vertices that are involved in the upcoming r operations (i.e., are endpoints of the edges to be inserted, or belong to the pairs to be queried). We build the complete graph over these at most $2r$ vertices, with the approximate shortest-path distances as edge weights. Each phase can then be handled by r edge insertions/queries on this smaller graph in $O\left(T_{\text{dyn}}(2r, r, 1+\varepsilon)\right)$ time. The resulting approximation factor is at most $(1+\varepsilon)^2 = 1+\Theta(\varepsilon)$. Thus, for $q = n^2$, we get the following bound:

$$T_{\text{offline}}(n, n^2, 1+\Theta(\varepsilon)) \;=\; O\left(\frac{n^2}{r} \cdot \left((1/\varepsilon)^5 n \log^3 n + r^2 + T_{\text{dyn}}(2r, r, 1+\varepsilon)\right)\right). \tag{1}$$

To solve Subproblem 1, we could do nothing during insertions and, in each query, re-run Dijkstra's algorithm from scratch. Then we would have that $T_{\text{dyn}}(2r, r, 1+\varepsilon) = O(r^3)$ and, by setting $r = (1/\varepsilon)^{5/3} n^{1/3} \log n$, $T_{\text{offline}}(n, n^2, 1+\Theta(\varepsilon))$ would be truly subcubic, namely $O\left((1/\varepsilon)^{10/3} n^{8/3} \log^2 n\right)$.

Actually, by using fast matrix multiplication and additional techniques, we can establish a better bound on $T_{\text{offline}}(n, n^2, 1+\Theta(\varepsilon))$. Our idea is to recursively divide phases into subphases, as in the proof of the following lemma. Note that this lemma actually holds for general graphs (although (semi-)dynamic shortest paths have been extensively studied in the literature, we are unable to find a known specific result that we can directly invoke).

▶ **Lemma 14** (A Semi-Dynamic Approximate Distance Oracle)**.** *We can solve Subproblem 1 in $T_{\text{dyn}}(2r, r, 1+\Theta(\varepsilon)) = O\left((1/\varepsilon) r^\omega \log r \log \overline{W}\right)$ total time, where ω is the matrix multiplication exponent and $\overline{W}$ is an upper bound on the maximum (finite) shortest-path distances.*

Proof. Let H be the input graph of $2r$ vertices, and let H' be the graph that results from performing to H all edge insertions of the first $r/2$ operations. We run the $O\left((1/\varepsilon) r^\omega \log \overline{W}\right)$-time $(1+\varepsilon)$-approximate APSP algorithm of Zwick [29] on H and H' and answer all distance queries therein. Then, we construct two graphs H_1 and H_2 of r vertices each, where H_1 (resp. H_2) is the complete graph over all vertices that are involved in the first (resp. last) $r/2$ operations; we set each edge weight in H_1 (resp. H_2) to be a $(1+\varepsilon)$-approximation of the shortest-path distance of its endpoints in H (resp. H') (which we have already computed), increasing thus the error by a $1+\varepsilon$ factor. Finally, we recurse in H_1 and H_2.

The running time of our approach is $T_{\text{dyn}}(2r, r, (1+\varepsilon)^i) \leq 2T_{\text{dyn}}(r, r/2, (1+\varepsilon)^{i+1}) + O\left((1/\varepsilon) r^\omega \log \overline{W}\right)$, where, initially, and $i = 1$; thus, we have that $T_{\text{dyn}}(2r, r, (1+\varepsilon)) = O\left((1/\varepsilon) r^\omega \log r \log \overline{W}\right)$. The approximation factor is $(1+\varepsilon)^{\log r} = 1+\Theta(\varepsilon \log r)$, which can be refined to $1+\varepsilon$, by resetting $\varepsilon \leftarrow \varepsilon/\log r$. ◀

Combining (1) with the above lemma gives

$$T_{\text{offline}}(n, n^2, 1+\Theta(\varepsilon)) \;=\; O\left(\frac{n^2}{r}\left((1/\varepsilon)^5 n \log^3 n + (1/\varepsilon) r^\omega \log r \log \overline{W}\right)\right).$$

Setting $r = n^{1/\omega}$ yields $T_{\text{offline}}(n, n^2, 1+\Theta(\varepsilon)) = O\left((1/\varepsilon)^5 n^{3-1/\omega} \log^3(n\overline{W})\right)$, where $\overline{W} \leq nW$, and W is the spread of S.

▶ **Theorem 15** (Approximate apBLSP). *Given a set S of n planar points of spread W, we can construct a data structure for the $(1+\varepsilon)$-approximate apBLSP problem of $O\left((1/\varepsilon)n^2 \log n\right)$ space, $O\left(\log\log n + \log(1/\varepsilon)\right)$ query, and $O\left((1/\varepsilon)^6 n^{3-1/\omega} \log^5(nW)\right)$ preprocessing time.*

The current best bound on the matrix multiplication exponent [26, 20] is $\omega < 2.373$, which gives a preprocessing time of $O\left((1/\varepsilon)^6 n^{2.579} \log^5(nW)\right)$.

Remark. For the sake of simplicity, we did not optimize the $\text{poly}(1/\varepsilon, \log(nW))$ factors. Specifically, it might be possible to avoid the dependency on the spread W by using known techniques, such as balanced quadtrees.

References

1 Srinivasa Arikati, Danny Z. Chen, L. Paul Chew, Gautam Das, Michiel Smid, and Christos D. Zaroliagis. Planar spanners and approximate shortest path queries among obstacles in the plane. In *Proceedings of the 4th Annual European Symposium on Algorithms (ESA)*, pages 514–528, 1996.

2 Prosenjit Bose, Anil Maheshwari, Giri Narasimhan, Michiel Smid, and Norbert Zeh. Approximating geometric bottleneck shortest paths. *Computational Geometry*, 29(3):233–249, 2004.

3 Costas Busch, Ryan LaFortune, and Srikanta Tirthapura. Improved sparse covers for graphs excluding a fixed minor. In *Proceedings of the 26th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 61–70, 2007.

4 Sergio Cabello. Many distances in planar graphs. *Algorithmica*, 62(1-2):361–381, 2012. doi:10.1007/s00453-010-9459-0.

5 Sergio Cabello. Subquadratic algorithms for the diameter and the sum of pairwise distances in planar graphs. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2143–2152, 2017.

6 Sergio Cabello and Miha Jejčič. Shortest paths in intersection graphs of unit disks. *Computational Geometry*, 48(4):360–367, 2015.

7 Paul B. Callahan and S. Rao Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *Journal of the ACM*, 42(1):67–90, 1995. doi:10.1145/200836.200853.

8 Timothy M. Chan. A dynamic data structure for 3-d convex hulls and 2-d nearest neighbor queries. *Journal of the ACM*, 57(3):16:1–16:15, 2010. doi:10.1145/1706591.1706596.

9 Timothy M. Chan and Dimitrios Skrepetos. All-pairs shortest paths in unit-disk graphs in slightly subquadratic time. In *Proceedings of the 27th Annual International Symposium on Algorithms and Computation (ISAAC)*, pages 24:1–24:13, 2016.

10 Timothy M. Chan and Dimitrios Skrepetos. Faster approximate diameter and distance oracles in planar graphs. In *Proceedings of the 25th European Symposium on Algorithms (ESA)*, pages 25:1–25:13, 2017.

11 Vincent Cohen-Addad, Søren Dahlgaard, and Christian Wulff-Nilsen. Fast and compact exact distance oracle for planar graphs. In *Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 962–973, 2017. doi:10.1109/FOCS.2017.93.

12 David Eppstein, Gary L. Miller, and Shang-Hua Teng. A deterministic linear time algorithm for geometric separators and its applications. *Fundamenta Informaticae*, 22(4):309–329, 1995.

13 Jie Gao and Li Zhang. Well-separated pair decomposition for the unit-disk graph metric and its applications. *SIAM Journal on Computing*, 35(1):151–169, 2005. Preliminary version in STOC 2003.

14 Pawel Gawrychowski, Shay Mozes, Oren Weimann, and Christian Wulff-Nilsen. Better tradeoffs for exact distance oracles in planar graphs. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 515–529, 2018.

15 Pawel, Gawrychowski, Haim Kaplan, Shay Mozes, Micha Sharir, and Oren Weimann. Voronoi diagrams on planar graphs, and computing the diameter in deterministic $\tilde{O}(n^{5/3})$ time. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 495–514, 2018.

16 Qian-Ping Gu and Gengchun Xu. Constant query time $(1+\varepsilon)$-approximate distance oracle for planar graphs. In *Proceedings of the 26th International Symposium on Algorithms and Computation (ISAAC)*, pages 625–636, 2015.

17 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 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2495–2504, 2017.

18 Ken-ichi Kawarabayashi, Christian Sommer, and Mikkel Thorup. More compact oracles for approximate distances in undirected planar graphs. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 550–563, 2013.

19 Philip Klein. Preprocessing an undirected planar network to enable fast approximate distance queries. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 820–827, 2002.

20 François Le Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 296–303, 2014. doi:10.1145/2608628.2608664.

21 Xiang-Yang Li, Gruia Calinescu, and Peng-Jun Wan. Distributed construction of a planar spanner and routing for ad hoc wireless networks. In *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, volume 3, pages 1268–1277, 2002.

22 Gary L Miller, S-H Teng, and Stephen A Vavasis. A unified geometric approach to graph separators. In *Proceedings of the 32nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 538–547, 1991.

23 Franco P. Preparata and Michael Ian Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, 1985.

24 Liam Roditty and Michael Segal. On bounded leg shortest paths problems. *Algorithmica*, 59(4):583–600, 2011. Preliminary version in SODA 2007.

25 Mikkel Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *Journal of the ACM*, 51(6):993–1024, 2004.

26 Virginia Vassilevska Williams. Multiplying matrices faster than Coppersmith–Winograd. In *Proceedings of the 44th ACM Symposium on Theory of Computing (STOC)*, pages 887–898, 2012. doi:10.1145/2213977.2214056.

27 Oren Weimann and Raphael Yuster. Approximating the diameter of planar graphs in near linear time. *ACM Transactions on Algorithms*, 12(1):1–13, 2016.

28 Chenyu Yan, Yang Xiang, and Feodor F. Dragan. Compact and low delay routing labeling scheme for unit disk graphs. *Computational Geometry*, 45(7):305–325, 2012. doi:10.1016/j.comgeo.2012.01.015.

29 Uri Zwick. All pairs shortest paths using bridging sets and rectangular matrix multiplication. *Journal of the ACM*, 49(3):289–317, 2002.

Dynamic Planar Orthogonal Point Location in Sublogarithmic Time

Timothy M. Chan
Department of Computer Science, University of Illinois at Urbana-Champaign, USA
tmc@illinois.edu

Konstantinos Tsakalidis[1]
Tandon School of Engineering, New York University, USA
kt79@nyu.edu

Abstract

We study a longstanding problem in computational geometry: dynamic 2-d orthogonal point location, i.e., vertical ray shooting among n horizontal line segments. We present a data structure achieving $O\left(\frac{\log n}{\log\log n}\right)$ optimal expected query time and $O\left(\log^{1/2+\varepsilon} n\right)$ update time (amortized) in the word-RAM model for any constant $\varepsilon > 0$, under the assumption that the x-coordinates are integers bounded polynomially in n. This substantially improves previous results of Giyora and Kaplan [SODA 2007] and Blelloch [SODA 2008] with $O(\log n)$ query and update time, and of Nekrich (2010) with $O\left(\frac{\log n}{\log\log n}\right)$ query time and $O\left(\log^{1+\varepsilon} n\right)$ update time. Our result matches the best known upper bound for simpler problems such as dynamic 2-d dominance range searching.

We also obtain similar bounds for orthogonal line segment intersection reporting queries, vertical ray stabbing, and vertical stabbing-max, improving previous bounds, respectively, of Blelloch [SODA 2008] and Mortensen [SODA 2003], of Tao (2014), and of Agarwal, Arge, and Yi [SODA 2005] and Nekrich [ISAAC 2011].

2012 ACM Subject Classification Theory of computation → Computational geometry, Theory of computation → Data structures design and analysis

Keywords and phrases dynamic data structures, point location, word RAM

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.25

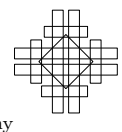
Acknowledgements We would like to thank Athanasios Tsakalidis for helpful discussions.

1 Introduction

Point location is one of the most well-studied and fundamental data structure problems in computational geometry. The static version of the problem dates back to the early years of the field. The *dynamic* version, which supports updates, has also received considerable attention, though obtaining $O(\log n)$ query and update time in 2-d remains open to this day; see [9] for the most recent breakthrough (and the extensive history).

There are two common formulations of 2-d point location. In the first, we want to store a connected planar subdivision with n edges so that we can quickly determine (the label of the) region containing any given query point q; updates correspond to insertions and deletions of edges. In the second formulation, also known as *vertical ray shooting*, we want to store a set of n disjoint line segments so that we can quickly report the lowest segment above

[1] Partially supported by NSF grants CCF-1319648 and CCF-1533564.



any given query point q; updates correspond to insertions and deletions of segments. Since knowing the segment immediately above q allows us to infer the region containing q, the first formulation reduces to the second, at least in the static case.[2] As in many other papers in this area, we focus only on the second formulation, which in some sense is more general since the segments do not need to be connected.

In this paper, we are interested in the *orthogonal* setting of the problem: in the vertical ray shooting formulation, the requirement is that all input line segments are horizontal. This case is among the most basic and important since many applications require handling only orthogonal input. The case when segments have a constant number of different slopes can also be reduced to the orthogonal case, due to the decomposability of vertical ray shooting (we can treat each slope class separately and apply a shear transform to each class).

Classical segment trees can solve the dynamic orthogonal problem with $O(\log^2 n)$ query and update time. In the late 1980s, Mehlhorn and Näher [18] improved the query and update bounds to $O(\log n \log\log n)$ by dynamic fractional cascading. At SODA'07 and SODA'08 respectively, Giyora and Kaplan [16] and Blelloch [6] both obtained $O(\log n)$ query and update time. Earlier at SODA'03, Mortensen [19] obtained $O(\log n)$ query and update time for the *decision* version of vertical ray shooting, namely, *vertical segment intersection emptiness* – deciding whether a vertical query segment intersects any of the horizontal input segments. These results are in the standard $(\log n)$-bit RAM model.

Sublogarithmic time? However, logarithmic time bounds are not the end of the story in the RAM model. For example, for the static orthogonal problem, Chan [8] at SODA'11 presented a linear-space data structure with $O(\log\log N)$ time if both x- and y-coordinates are integers bounded by N.

For the dynamic orthogonal problem, Alstrup et al. [2] applied Fredman and Saks' lower-bound technique [15] to show that any data structure with t_u update time requires $\Omega\left(\frac{\log n}{\log(t_u \log n)}\right)$ query time in the cell-probe model (with $(\log n)$-bit cells). In particular, any data structure with polylogarithmic update time requires $\Omega\left(\frac{\log n}{\log\log n}\right)$ query time. Nekrich [23] has shown that $O\left(\frac{\log n}{\log\log n}\right)$ query time is possible with a data structure for dynamic 2-d orthogonal point location supporting $O\left(\log^{1+\varepsilon} n\right)$ update time. But could we obtain $O\left(\frac{\log n}{\log\log n}\right)$ optimal query time and $O\left(\frac{\log n}{\log\log n}\right)$ update time? Or more ambitiously, could we obtain the same optimal query time with substantially sublogarithmic update time? Alstrup et al.'s lower bound does not rule out this possibility.

Indeed, a phenomenon of "fractional-power-of-log" update times has been observed for several problems with Fredman–Saks-style lower bounds. For example, for dynamic 1-d *rank* queries (or equivalently, 1-d range counting) and *selection* queries, Chan and Pătraşcu [10] obtained a data structure with $O\left(\frac{\log n}{\log\log n}\right)$ optimal query time and $O\left(\log^{1/2+\varepsilon} n\right)$ amortized update time, where $\varepsilon > 0$ denotes an arbitrarily small constant. For dynamic 2-d *orthogonal range searching*, Mortensen in 2006 [21] gave a data structure with $O\left(\frac{\log n}{\log\log n}\right)$ optimal query time and $O\left(\log^{5/6+\varepsilon} n\right)$ amortized update time in the special case of 3-sided queries, or $O\left(\log^{7/8+\varepsilon} n\right)$ amortized update time in general. Wilkinson in 2014 [29] improved

[2] The reduction also holds in the dynamic case, by storing the edges around each region in an "ordered union-split-find" structure [17]. Unfortunately, such a structure requires logarithmic overhead and is inadequate for us, as we aim for sublogarithmic bounds.

Mortensen's update time in the 3-sided case to $O\left(\log^{2/3+\varepsilon} n\right)$, and obtained $O\left(\log^{1/2+\varepsilon} n\right)$ update time in the 2-sided case. Our SoCG paper last year [12] improved these update times further, to $O\left(\log^{1/2+\varepsilon} n\right)$ in the 3-sided case, and $O\left(\log^{2/3+o(1)} n\right)$ in general, while retaining $O\left(\frac{\log n}{\log\log n}\right)$ optimal query time.

Orthogonal range searching and vertical ray shooting are related: 3-sided orthogonal range searching is equivalent to the *1-sided* special case of vertical ray shooting where all input segments are horizontal rays. Mortensen's Ph.D. thesis [20] combined both his papers on range searching [21] and segment intersection emptiness/reporting [19], and it is interesting to note that he was able to obtain fractional-power-of-log update times for the former but not for the latter in general, suggesting that dynamic 2-d orthogonal point location might be a tougher problem than dynamic 2-d orthogonal range searching.

New result. We succeed in simultanesouly obtaining sublogarithmic query time and substantially sublogarithmic update time for dynamic 2-d orthogonal point location (in the vertical ray shooting formulation), under the assumption that x-coordinates are integers bounded polynomially in n. Our data structure achieves $O\left(\frac{\log n}{\log\log n}\right)$ optimal query time and $O\left(\log^{1/2+\varepsilon} n\right)$ update time, greatly improving the previous results of Giyora and Kaplan [16], Blelloch [6], Mortensen [19], and Nekrich [23]. Our results are in the word-RAM model, under the standard assumption that the word size w is at least $\log n$ bits (in fact, except for an initial predecessor search during each query/update, we only need operations on $(\log n)$-bit words). Both our query and update bounds are amortized. Our query time bound is expected: randomization is used, but only in the query algorithm, not in the update algorithm. For the decision problem, vertical segment intersection emptiness, our algorithm is completely deterministic. Our algorithm can be extended to solve vertical segment intersection reporting – reporting all horizontal input segments intersecting a vertical query segment – in $O\left(\frac{\log n}{\log\log n} + k\right)$ deterministic time where k is the number of output segments.

Interestingly, our update time bound is even better than our earlier $O\left(\log^{2/3+o(1)} n\right)$ result [12] for general 2-d orthogonal range searching. There are reasons to believe that $\log^{1/2+\varepsilon} n$ could be the best possible, under current knowledge: The current best result for the simpler problem of 2-d orthogonal *2-sided* (i.e., *dominance*) range searching by Wilkinson [29] already has $O\left(\log^{1/2+\varepsilon} n\right)$ update time; this simpler problem corresponds to the special case of 2-d orthogonal point location where all input line segments and query line segments are rays. Any improvement of our update time would require improvement in this special case first. Besides, sub-$\sqrt{\log n}$ update upper bounds have never been obtained before for *any* problem with Fredman–Saks-style lower bounds.

The assumption of a polynomially bounded x-universe is reasonable and holds in most applications. For example, in offline settings where we know the coordinates of all the input segments in advance, we can simply replace coordinates by their ranks. The known lower bounds still hold in the bounded universe setting. The assumption arises from a technical issue in ensuring balance in our underlying tree structure. It is plausible that it could be eliminated with more technical effort, but we would rather prefer keeping the solution simpler.

Overview of techniques. Following our earlier approach [12] for dynamic orthogonal range searching (which in turn was a simplification of the approach of Mortensen [21]), our general strategy consists of three parts:

1. We first design efficient *micro-structures* for solving the problem for small input of size s, with the goal of achieving near-constant amortized update time. Following [12, 29], this part is obtained by taking an external-memory version of the segment tree, and re-implementing it in internal memory using bit-packing tricks. We use ideas from *buffer trees* [4] to aim for an amortized update cost of the form $O((\log s)/B)$, where B is the block size. Since we can pack $B \approx (\log s)/w$ segments in one word in internal memory, this would yield update time $O((\log^2 s)/w)$, which is indeed small when $s \approx 2^{\sqrt{\log n}}$.
2. Next we devise a black-box transformation to convert micro-structures into data structures for the intermediate case when input coordinates lie in a narrow $s \times n$ grid. Following Mortensen [21, 19], this part can be obtained from a $\sqrt{n}$-way divide-and-conquer similar to van Emde Boas trees [27]. (This part does not require bit packing.) The query and update time increase only by a $\log\log n$ factor as a result of this "van Emde Boas transformation".
3. Finally we give another black-box transformation to convert a narrow-grid data structure into a *macro-structure* for the general problem for large input. This part is obtained by using a global segment tree with fan-out near s. (This part does not require bit packing either.) The update time increases by a factor of $\log_s n$ (the height of the tree), which becomes near $\sqrt{\log n}$.

While this high-level plan may appear similar to our previous paper [12], at least two major difficulties arise, if we want the best update and query time: First, in part 1, buffered segment trees for 2-d orthogonal point location actually require $O((\log^2 s)/B) = O((\log^3 s)/w)$ update time, which forces us to set $s \approx 2^{\log^{1/3} n}$ and leads to a worse final update time near $\log_s n \approx \log^{2/3} n$. Second, the van Emde Boas transformation in part 2 causes at least one extra $\log\log n$ factor and leads to suboptimal query time in the end. A number of new ideas are thus needed to deal with these difficulties.

To overcome the first obstacle, our idea is to use micro-structures only for the simpler 1-sided case where input segments are horizontal rays, for which $O((\log s)/B)$ update cost is indeed possible. But how can we avoid using micro-structures for general 2-sided segments in part 3? We observe that an input segment appears more often as 1-sided than 2-sided at the nodes of segment tree, so we can afford to handle 2-sided updates by switching to a slower algorithm, with bootstrapping.

To overcome the second obstacle, we suggest a new van Emde Boas transformation to trade off the $\log\log n$ increase in the query time with a $\log^{\varepsilon} n$ increase in the update time. We can obtain such a trade-off only for the decision version of the problem, but luckily there are known randomized techniques [7] to reduce the original problem to the decision problem without hurting the expected query time.

As a by-product of our new van Emde Boas transformation, we can immediately obtain a data structure for *dynamic 1-d range emptiness* with $O(1)$ query time and $O(\log^{\varepsilon} N)$ update time for an integer universe bounded by N. This bound was known before: Mortensen, Pagh and Pătraşcu [22] in fact provided optimal results for a complete query/update time trade-off, but our solution in the constant query-time case is simpler, and may be of independent interest (if it has not been observed before).

Applications. Our result improves previous results even in various special cases:

- Dynamic *vertical ray stabbing* refers to the special case of vertical segment intersection reporting where the query segments are vertical rays. The problem has applications in databases, GIS, and networking. Previously, only logarithmic query and update time were known [26].

- Dynamic *vertical stabbing-min* refers to special case of vertical ray shooting where the query point has y-coordinate at $-\infty$ (stabbing-max is symmetric). Previously, Agarwal, Arge, and Yi at SODA'05 [1] obtained logarithmic query and update time. More recently, Nekrich [24] obtained $O\left(\frac{\log n}{\log\log n}\right)$ query and $O(\log n)$ update time; our $O\left(\log^{1/2+\varepsilon} n\right)$ update time is a significant improvement.

To further illustrate how fundamental our results are, we mention two offline applications (where as mentioned, the polynomially bounded universe assumption can automatically be ensured by sorting and replacing coordinates with ranks). Both applications are interesting in their own right.

- An $O\left(n\log^{1/2+\varepsilon} n\right)$-space data structure with $O(\log n)$ expected query time for static *3-d vertical ray shooting*: store a set of n axis-aligned rectangles in 3-d parallel to the xy-plane, so that we can find the lowest rectangle above a query point. This problem can be viewed as a variant of 3-d orthogonal point location. Our space bound is unusual and intriguing. The result can be immediately obtained by using the standard sweep-plane algorithm, together with a (partially) persistent version of our dynamic data structures for 2-d vertical ray shooting. The space usage is proportional to the total time to process n insertions and n deletions, which is $O\left(n\log^{1/2+\varepsilon} n\right)$; using Dietz's persistent arrays [14], the query time increases by a $\log\log n$ factor, to $O\left(\frac{\log n}{\log\log n}\log\log n\right) = O(\log n)$. This improves a previous $O\left(n\log^{1+\varepsilon} n\right)$-space data structure with $O(\log n)$ query time [8, Corollary 4.2(e)].
- A deterministic $O\left(n\frac{\log n}{\log\log n}\right)$-time algorithm for *single-source shortest paths in an unweighted intersection graph of n axis-aligned line segments* in 2-d, e.g., finding a path between two points with the minimum number of turns in an arrangement of vertical and horizontal line segments ("roads"). Recently, Chan and Skrepetos [11] described an $O(n\log n)$-time algorithm by simulating breadth-first search using a dynamic data structure for orthogonal segment intersection emptiness, performing at most n queries and n deletions. Our new data structure immediately improves the total running time to $O\left(n\log^{1/2+\varepsilon} n + n\frac{\log n}{\log\log n}\right) = O\left(n\frac{\log n}{\log\log n}\right)$.

2 Preliminaries

In all our algorithms, we assume that during each query or update, we are given a pointer to the predecessor/successor of the x- and y- values of the given point or segment. At the end, we can add the cost of predecessor search to the query and update time (which is no bigger than $O\left(\sqrt{\log n}\right)$) [3] in the word RAM model, or $O(\log\log n)$ in the polynomial universe case [27]).

We assume a word RAM model that allows for a constant number of non-standard operations on w-bit words. By setting $w := \delta\log n$ for a sufficiently small constant δ, these operations can be simulated in constant time by table lookup, after preprocessing the tables in $2^{O(w)} = n^{O(\delta)}$ time.

For most of the paper, we focus on solving the decision problem, i.e., vertical segment intersection emptiness. Vertical ray shooting will be addressed in Section 7 afterwards. Adapting our algorithm for segment intersection reporting will be straightforward.

Let $[n]$ denote $\{0, 1, \ldots, n-1\}$.

We now quickly review a few useful tools (also used in our previous paper [12]).

Weight-balancing. *Weight-balanced B-trees* [5] are B-tree implementations with a rebalancing scheme that is based on the nodes' *weights*, i.e., subtree sizes, in order to support updates of secondary structures efficiently.

▶ **Lemma 1** ([5, Lemma 4]). *In a weight-balanced B-tree of degree r, nodes at height i have weight $\Theta\left(r^i\right)$, and any sequence of n insertions requires at most $O\left(n/r^i\right)$ splits of nodes at height i.*

(We do not need to address balancing after deletions, since we can handle deletions lazily, and rebuild periodically when the size of the tree decreases or increases by a constant factor [5, 25].)

Colored predecessors. *Colored predecessor searching* is the problem of maintaining a dynamic set of multi-colored, totally ordered elements and searching for the predecessors with a given color.

▶ **Lemma 2** ([21, Theorem 14]). *Colored predecessor searches and updates on n colored, totally ordered elements can be supported in $O\left(\log^2 \log n\right)$ time deterministically.*

Initial structure. For bootstrapping purposes, we need an initial data structure for vertical segment intersection emptiness with optimal $O\left(\log_w n\right)$ query time, allowing possibly large but polylogarithmic update time. Nekrich [23] has already given such a structure with $O\left(\log^{1+\varepsilon} n\right)$ update time. We state a weaker bound, which is sufficient for our purposes (and is simpler to obtain):

▶ **Lemma 3.** *For n horizontal segments in the plane, there exists a dynamic data structure for vertical segment intersection emptiness that support updates in $O\left(\log^{2+\varepsilon} n\right)$ amortized time and queries in $O\left(\log_w n\right)$ time.*

3 Micro-structures

In this section, we consider *micro-structures* for vertical segment intersection emptiness when the number of input segments s is small. This part relies on bit packing techniques. We focus on the *1-sided* special case, when all the input segments are horizontal rays. Without loss of generality, we may assume that all rays are rightward (since we can treat leftward rays separately). Vertical segment intersection emptiness in the 1-sided case is equivalent to 2-d *3-sided orthogonal range emptiness*: store a set of input points so that we can quickly decide whether a query 3-sided rectangle contains any input point. To see the equivalence, just replace the input rays with their endpoints, and each vertical query segment $\{x\} \times [y_1, y_2]$ with the 3-sided rectangle $(-\infty, x] \times [y_1, y_2]$.

We can adapt our previous micro-structures for 3-sided orthogonal range searching [12] to obtain:

▶ **Lemma 4.** *Let $b \geq 2$ be an integer. Given s horizontal (1-sided) rays with endpoints from $[s] \times \mathbb{R}$, there exists a dynamic data structure for vertical segment intersection emptiness with the following amortized update and query time:*

$$\begin{aligned} U_1(s) &= O\left(\frac{b \log^2 s}{w} + b \log^2 \log s\right) \\ Q_1(s) &= O\left(\log_b s\right). \end{aligned}$$

Proof. The case $b = 2$ has already been proved in [12, Lemmata 5(i) and 6]. We only briefly review the proof outline, to note the easy generalization to arbitrary b.

First consider the case when the endpoints all come from a static universe $[s] \times [s]$. The idea is to mimick an existing external-memory data structure with a block size of $B := \lceil \frac{\delta w}{\log s} \rceil$, observing that B points can be packed in a single word, assuming a sufficiently small constant δ. For such a external-memory structure, Chan and Tsakalidis [12] chose a *buffered* version [4] of a binary *priority search tree* ordered by y, citing Wilkinson [29]. Here, we use a buffered b*-ary* priority search tree instead, which according to Wilkinson [29, Lemma 1] has $O\left(b \cdot \frac{1}{B} \cdot \log s + 1\right) = O\left(\frac{b \log^2 s}{w} + 1\right)$ amortized update time and $O\left(\log_b s\right)$ amortized query time.

To make this data structure support a dynamic y-universe, Chan and Tsakalidis [12] applied monotone list labeling techniques; the extra update cost is $O(\log^2 \log s)$. It is straightforward to check that the same approach works in the b-ary variant, with an extra overhead factor of $O(b)$. ◀

Note that the first term in the above update time is constant when the number of segments s is bounded by $2^{\sqrt{w}}$.

We could also consider micro-structures for vertical segment intersection emptiness in the general (2-sided) case, but they seem to require more than two $\log s$ factors in the update time (not to mention possibly worse query time), which would result in a final update time worse than $\sqrt{\log n}$. Luckily, our macro-structures later need micro-structures only for the 1-sided case.

4 Van Emde Boas transformation

Next, we consider *macro-structures* for vertical segment intersection emptiness when the number of input segments n is large. As an intermediate step, we first adapt a technique of Mortensen [19, 21] that transforms any micro-structure for vertical segment intersection emptiness on s horizontal segments to one for n segments with endpoints from a narrow grid $[s] \times \mathbb{R}$.

The transformation uses a recursion similar to van Emde Boas trees [28], and increases both update and query time by a $\log \log n$ factor. Although the extra factor is small, we cannot afford to lose it if we want sublogarithmic query time at the end. We present a new variant of van Emde Boas recursion, with a parameter b, that allows us to trade off the query-time increase with an update-time increase:

▶ **Lemma 5.** *Let $b \geq 2$ be an integer. Given a dynamic data structure for vertical segment intersection emptiness on s horizontal segments with endpoints from $[s] \times \mathbb{R}$ achieving update time $U(s)$ and query time $Q(s)$, there exists a dynamic data structure for vertical segment intersection emptiness on n horizontal segments with endpoints from $[s] \times \mathbb{R}$ achieving the following update and query time:*

$$\begin{aligned} U(s,n) &= O\left(bU(s^5) \log^2 \log n + b \log^3 \log n\right) \\ Q(s,n) &= O\left(Q(s^5) \log_b \log n\right). \end{aligned}$$

An analogous transformation holds for the 1-sided special case of vertical segment intersection emptiness.

Proof. We present our variant of van Emde Boas recursion a little differently than usual, as a near-$n^{1/b}$-degree tree, with recursively defined secondary structures.

The data structure. We store a degree-r tree ordered by y, implemented as a weighted-balanced B-tree, for some parameter r to be chosen later. Each node corresponds to a horizontal slab; its slab is divided into its children's slabs by at most r dividing horizontal lines. We say that two input segments are in the same *class* if they have the same pair of left and right x-coordinates; there are at most s^2 classes. At each node v, we store the input segments in v's slab in one y-sorted list per class, in a colored predecessor search structure, and define the following lists:

1. Let $M(v)$ contain the bottommost and topmost segments (the "min" and the "max") in v's slab for each class. Since $|M(v)| \leq s^2$, we can maintain $M(v)$ in the given micro-structure supporting updates in $U(s^2)$ time and queries in $Q(s^2)$ time.
2. Let $R_0(v)$ contain the segments in v's slab after "rounding" down the y-coordinate to align with one of the r lines dividing the slab. Duplicates are removed from $R_0(v)$.
   Let $R(v)$ be equal to $R_0(v)$ but *excluding the bottommost and topmost rounded segment per class.* Since $R(v)$ has at most r distinct y-coordinates and at most $s^2 r$ segments, we can maintain $R(v)$ in a data structure supporting updates in $U(s, s^2 r)$ time and queries in $Q(s, s^2 r)$ time by recursion. (Note that we maintain $R(v)$, but not $R_0(v)$.) We further assume that this structure has $U_{\text{prep}}(s, s^2 r)$ *amortized preprocessing time*, i.e., preprocessing time divided by the number of input segments.

The update algorithm. To insert or delete a horizontal segment p, we proceed as follows:

1. Identify the path π of $O(\log_r n)$ nodes whose slabs contain p. Update the sorted lists for p's class at these nodes.
2. For each node $v \in \pi$ for which $M(v)$ changes, update the data structure for $M(v)$.
3. For each node $v \in \pi$ for which $R(v)$ changes, update the data structure for $R(v)$.

In step 1, the $O(\log_r n)$ sorted lists can be updated in $O\left(\log_r n \log^2 \log n\right)$ time by colored predecessor search (Lemma 2).

Step 2 takes at most $O(\log_r n)$ updates to the micro-structures and thus costs $O(\log_r n)\, U(s^2)$ time.

For step 3, we claim that $R(v)$ changes only at one node $v \in \pi$: specifically, the lowest node on π that contains at least one other segment of p's class. To see why, for any node $v' \in \pi$ strictly above v, there is no change to $R_0(v')$ (and thus $R(v')$) since there is another segment that gives the same rounded segment as p at v'; on the other hand, for any node $v' \in \pi$ strictly below v, there is no change to $R(v')$ because p is the only segment in its class at v' and would be excluded from $R(v')$.

Note that if $R(v)$ changes, it changes by at most a single insertion or deletion (for example, if the segment we are inserting becomes the new bottommost segment in a class, we insert the old bottommost segment to $R(v)$). Thus, step 3 takes $U(s, s^2 r)$ time.

To keep the tree balanced, we need to handle node splits. For nodes at height i, there are $O\left(n/r^i\right)$ splits by Lemma 1. A split at a non-leaf node v at height i requires rebuilding $M(v)$ and $R(v)$, which takes at most $O\left(r^i U(s^2) + r^i U_{\text{prep}}(s, s^2 r)\right)$ time. It also requires updating the y-coordinates of $O(s^2)$ segments in $R(v')$ at the parent v' of v, which takes $O\left(s^2\, U(s, s^2 r)\right)$ time. The total extra cost is at most

$$O\left(\sum_{i=1}^{\log_r n} (n/r^i) \cdot \left[r^i U(s^2) + r^i U_{\text{prep}}(s, s^2 r) + s^2\, U(s, s^2 r)\right]\right)$$
$$= O\left(n \cdot \left[U(s^2) \log_r n + U_{\text{prep}}(s, s^2 r) \log_r n + \frac{s^2}{r} U(s, s^2 r)\right]\right).$$

To summarize, we obtain the following recurrence for the amortized update time:

$$\begin{aligned} U(s,n) \;\leq\; & U\left(s, s^2 r\right) + O\left(\log_r n\right) U\left(s^2\right) + \\ & O\left(U_{\text{prep}}(s, s^2 r)\log_r n + \frac{s^2}{r} U(s, s^2 r) + \log_r n \log^2 \log n\right). \end{aligned}$$

The amortized preprocessing time is given by the following simpler recurrence, since balance is easily ensured at preprocessing:

$$U_{\text{prep}}(s,n) \;\leq\; U_{\text{prep}}\left(s, s^2 r\right) + O\left(\log_r n\right) U\left(s^2\right) + O\left(\log_r n \log^2 \log n\right).$$

The query algorithm. To answer a query for a vertical segment q with bottom endpoint q_1 and top endpoint q_2, we proceed as follows:

1. Find the lowest node v whose slab contains both q_1 and q_2 by performing an LCA query for the two leaves containing them.
2. Let v_1 and v_2 be the two children of v whose slabs contain q_1 and q_2.
3. Answer the query in $M(v_1)$, $M(v_2)$, and $M(v)$. Also, round q_1 upward and q_2 downward, then answer the query in $R(v)$. Return true iff one of these queries returns true.

To prove correctness, suppose that q intersects the horizontal input segment p. If p is in v_1's slab, then q intersects also the topmost segment in v_1 of p's class and so the query in $M(v_1)$ would return yes. If p is in v_2's slab, then similarly the query in $M(v_2)$ would return yes. If p is in neither slab, then q intersects the segment p after rounding and so the query in $R(v)$ would return yes, unless p after rounding is the topmost or bottommost rounded segment in v of its class. In this exceptional case, p would be excluded from $R(v)$, but then the query in $M(v)$ would return yes.

Since LCA queries take $O(1)$ time (with $O(1)$ update time) [13], we obtain the following recurrence for the query time:

$$Q(s,n) \;\leq\; Q\left(s, s^2 r\right) + O\left(Q\left(s^2\right)\right).$$

Conclusion. We set $r := s^2 n^{1/b}$ to obtain

$$\begin{aligned} U_{\text{prep}}(s,n) &\leq U_{\text{prep}}\left(s, s^4 n^{1/b}\right) + O\left(bU\left(s^2\right) + b\log^2 \log n\right) \\ U(s,n) &\leq \left(1 + \frac{1}{n^{1/b}}\right) U\left(s, s^4 n^{1/b}\right) + O\left(bU\left(s^2\right) + bU_{\text{prep}}(s,n) + b\log^2 \log n\right) \\ Q(s,n) &\leq Q\left(s, s^4 n^{1/b}\right) + O\left(Q\left(s^2\right)\right). \end{aligned}$$

For the base case, we can use $U_{\text{prep}}(s, s^5), U(s, s^5) = O(U(s^5))$ and $Q(s, s^5) = O(Q(s^5))$. The recurrences solve to $U_{\text{prep}}(s,n) = O\left(bU(s^5)\log\log n + b\log^2 \log n\right)$, $U(s,n) = O\left(b^2 U(s^5)\log^2 \log n + b^2 \log^3 \log n\right)$, and $Q(s,n) = O(Q(s^5)\log_b \log n)$. Resetting $b \leftarrow \lceil \sqrt{b} \rceil$ yields the lemma. ◀

Remark. The above approach gives a data structure for dynamic *1-d range emptiness* (which corresponds to the special case of $s = 1$) with $O(b \log\log N)$ update time and $O(\log_b \log N)$ query time in an integer universe $[N]$. (We do divide-and-conquer to reduce the universe size N rather than the number of points n; balancing is no longer an issue, so the extra $\log\log$ factor in the update bound goes away.) In particular, setting $b = \log^{\varepsilon} N$ gives $O(1)$ query time and $O(\log^{O(\varepsilon)} N)$ update time. This result was known before by Mortensen, Pagh, and Pătraşcu [22], who gave a more complicated method achieving a better (optimal) trade-off,

with $O(b \log\log N)$ update time and $O(\log\log_b \log N)$ query time, and also with $O(n)$ space. However, it is interesting to note that the above variant of van Emde Boas tree is sufficient for the constant-query-time case.

5 Segment tree transformation

We next describe macro-structures to transform data structures for vertical segment intersection emptiness for n segments in the narrow grid case, to the general case. The transformation is based on a multi-degree segment tree (the idea is standard and, for example, was used in Giyora and Kaplan's paper [16]).

▶ **Lemma 6.** *Given a dynamic data structure for vertical segment intersection emptiness on n horizontal segments with endpoints from $[s] \times \mathbb{R}$ achieving update time $U(s,n)$ and query time $Q(s,n)$, and a dynamic data structure for vertical segment intersection emptiness on n horizontal (1-sided) rays with endpoints from $[s] \times \mathbb{R}$ achieving update time $U_1(s,n)$ and query time $Q_1(s,n)$, there exists a data structure for dynamic vertical segment intersection emptiness on n horizontal segments with endpoints from $[N] \times \mathbb{R}$ achieving the following amortized update and query time:*

$$\begin{aligned} U(N,n) &= O\left(U_1(s,n)\log_s N + U(s,n) + \log_s N \log^2\log n\right) \\ Q(N,n) &= O\left((Q_1(s,n) + Q(s,n))\log_s N\right). \end{aligned}$$

Proof. We store a degree-s segment tree ordered by x, with N leaves and height $O(\log_s N)$; each node corresponds to a vertical slab.

We describe how each input segment p is stored. Let v be the lowest node that contains both endpoints of p, i.e., the LCA of the two leaves containing the endpoints. Let v_ℓ and v_r be the two children of v whose slabs contain the two endpoints. We divide p into three *subsegments*: the *left* and *right subsegments*, i.e., the parts of p within the slabs of v_ℓ and v_r respectively, and the remaining *middle subsegment*, i.e., the part within the union of the slabs of the children of v strictly between v_ℓ to v_r.

We store the middle subsegment of p in a data structure on the narrow grid $X_v \times \mathbb{R}$, where X_v is the set of x-coordinates of the $O(s)$ dividing vertical lines at node v.

We store the left subsegment of p along a path of $O(\log_s N)$ nodes. We first find the child v'_ℓ of v_ℓ whose slab contains the left endpoint of p. We divide the left subsegment into: the *left left subsegment*, i.e., the part within the slab of v'_ℓ, and the remaining *left middle subsegment*. We store the left middle subsegment in a 1-sided data structure on the narrow grid $X_{v'_\ell} \times \mathbb{R}$; note that this subsegment appears as a rightward ray in the narrow grid and so is indeed 1-sided. We then repeat for the left left subsegment in the slab at v'_ℓ.

We store the right subsegment of p symmetrically.

In addition, we store the y-coordinates of the segments/rays stored at each node v in a colored predecessor searching structure of Lemma 2, where segments/rays with endpoints in the same child's slab are assigned the same color. We also store the x-coordinates in another colored predecessor searching structure, where X_v is colored black and the rest is white.

To insert or delete a segment p, we insert or delete the middle subsegment in $O(U(s,n))$ time and the $O(\log_s N)$ pieces of the left/right subsegment in $O(U_1(s,n)\log_s N)$ time. Note that given the y-predecessor of the segment at a node v, we can obtain the y-predecessor/successor at the child by using the colored predecessor searching structure. We can also determine the x-predecessor/successor of its endpoints in X_v by another colored predecessor search. This takes total extra time $O\left(\log_s N \log^2\log n\right)$.

To answer an intersection emptiness query for a vertical segment q, we proceed down the path π of nodes whose slabs contain q, and perform queries in the narrow-grid structures (both general and 1-sided) at nodes in π. This takes $O((Q(s,n)+Q_1(s,n))\log_s N)$ time. ◀

6 Putting everything together

We can finally obtain our main result by combining with our preceding micro-structures and by bootstrapping.

▶ **Theorem 7.** *Given n horizontal segments with coordinates from $[N]\times\mathbb{R}$, there exists a dynamic data structure for vertical segment intersection emptiness that supports updates in amortized $O\left(\frac{\log N}{\log^{1/2-\varepsilon} n}+\log^{1/3} n\right)$ time and queries in $O\left(\frac{\log N}{\log\log n}\right)$ time if $N\geq n$.*

Proof. To make calculations more readable, we introduce the notation O^* to hide factors of the form $\log^{O(\varepsilon)} n$ and $w^{O(\varepsilon)}$.

Combining our van Emde Boas transformation in Lemma 5 with $b=\log^{\varepsilon} n$ and segment tree transformation in Lemma 6 gives

$$\begin{aligned} U(N,n) &= O^*\left(U_1(s)\log_s N+U(s)\right) \\ Q(N,n) &= O\left((Q_1(s)+Q(s))\log_s N\right). \end{aligned} \tag{1}$$

The 1-sided micro-structure in Lemma 4 with $b=w^{\varepsilon}$ gives $U_1(s^5)=O^*\left(\frac{\log^2 s}{w}+1\right)$ and $Q_1(s^5)=O(\log_w s)$. The initial structure in Lemma 3 gives $U(s^5)=O\left(\log^{2+\varepsilon} s\right)$ and $Q(s^5)=O(\log_w s)$. Thus,

$$\begin{aligned} U(N,n) &= O^*\left(\left(\frac{\log^2 s}{w}+1\right)\log_s N+\log^{2+\varepsilon} s\right) \\ Q(N,n) &= O(\log_w s\log_s N) = O(\log_w N). \end{aligned}$$

Setting $s=2^{\log^{1/3} N}$ then yields $U(N,n)=O^*\left(\log^{(2+\varepsilon)/3} N+\frac{\log^{4/3} N}{w}\right)$ and $Q(N,n)=O(\log_w N)$.

To improve the update time further, we bootstrap with our new bounds $U(s^5)=O^*\left(\log^{(2+\varepsilon)/3} s+\frac{\log^{4/3} s}{w}\right)$ and $Q(s^5)=O(\log_w s)$. Then

$$\begin{aligned} U(N,n) &= O^*\left(\left(\frac{\log^2 s}{w}+1\right)\log_s N+\log^{(2+\varepsilon)/3} s+\frac{\log^{4/3} s}{w}\right) \\ Q(N,n) &= O(\log_w s\log_s N) = O(\log_w N). \end{aligned} \tag{2}$$

Setting $s=2^{w^{1/2-\varepsilon}}$ yields $U(N,n)=O^*\left(\frac{\log N}{w^{1/2-\varepsilon}}+w^{1/3}\right)$ and $Q(N,n)=O(\log_w N)$. Setting the word size $w=\delta\log n$ gives the theorem. (The $\log^{1/3} n$ term could probably be lowered by further rounds of bootstrapping, but that term does not matter in the main case of interest, when $N=n^{O(1)}$.) ◀

7 Vertical ray shooting

We now extend our query algorithm for vertical segment intersection emptiness to vertical ray shooting.

Extended van Emde Boas transformation. First, we note a naive extension of the van Emde Boas transformation to support vertical ray shooting (the query time isn't optimized):

▶ **Lemma 8.** *The same data structure in Lemma 5 can answer vertical ray shooting queries in time* $\vec{Q}(s,n) = O\left(b\vec{Q}\left(s^5\right)\log\log n\right)$. *An analogous result holds for the 1-sided case.*

Proof. To answer a vertical ray shooting query for a point q, we proceed as follows:

1. Identify the path π of $O(\log_r n)$ nodes whose slabs contain q.
2. Find the lowest node $v \in \pi$ for which the answer of the query in $M(v)$ is nonempty.
3. Answer the query in $R(v)$ and in $M(v)$, and suppose that the two answers are in the slab of the children v_1 and v_2 of v respectively. Return the answer to the query in $M(v_1)$ or $M(v_2)$, whichever is lower.

To show correctness, let p be the lowest segment above q. After step 2, we know that p must be in the slab of v but not v's child in π. After step 3, we know that p must be in the slab of v_1 and in $M(v_1)$, unless p after rounding is the topmost or bottommost rounded segment in v of its class. In this exceptional case, p would be excluded from $R(v)$, but then p would be in the slab of v_2 and in $M(v_2)$.

We get the following recurrence for the query time:

$$\vec{Q}(s,n) \;\le\; \vec{Q}(s,s^2r) + O(\log_s n)\,\vec{Q}(s^2).$$

For $r = s^2n^{1/b}$, this gives $\vec{Q}(s,n) \le \vec{Q}(s,s^4n^{1/b}) + O\left(b\vec{Q}(s^2)\right)$, and the recurrence can be solved as before. ◀

Extended segment tree transformation. Next we extend the segment tree transformation. For this part, we will optimize the query time, using a randomized search technique from [7].

▶ **Lemma 9.** *In Lemma 6, if the given general and 1-sided data structures can answer vertical ray shooting queries in* $\vec{Q}(s,n)$ *and* $\vec{Q}_1(s,n)$ *time respectively, then the new data structure can answer vertical ray shooting queries in expected query time*

$$\vec{Q}(N,n) = O\left((Q_1(s,n)+Q(s,n))\log_s N + \left(\vec{Q}_1(s,n)+\vec{Q}(s,n)\right)\log\log_s N\right).$$

Proof. The technique [7] is based on the following simple well-known observation: the minimum of m unknown elements $y_1,\ldots,y_m$ can be found with m comparisons of the form "is $y_i < y$?" for a given value y, and $O(\log m)$ expected number of evaluations of the y_i's. (The observation follows by running the standard algorithm for the minimum, after randomly permuting the elements.)

To answer a vertical ray shooting query for a point q, we proceed down the path π of nodes whose slabs contain q, and apply the above observation with $m = O(\log_s N)$ and y_i representing the y-value of the lowest segment above q in the narrow-grid structures at the i-th node of π. Deciding "$y_i < y$?" is equivalent to performing a vertical segment emptiness query. The theorem follows. ◀

▶ **Theorem 10.** *The same data structure in Theorem 7 can answer vertical ray shooting queries in* $O\left(\frac{\log N}{\log\log n}\right)$ *time.*

Proof. By the above extensions, the combined van Emde Boas transformation (with $b = \log^\varepsilon n$) and segment tree transformation that yielded (1) has

$$\begin{aligned}\vec{Q}(N,n) &= O\left((Q_1(s)+Q(s))\log_s N + \left(\vec{Q}_1(s)+\vec{Q}(s)\right)\log^\varepsilon n\log\log n\log\log_s N\right)\\ &= O\left((Q_1(s)+Q(s))\log_s N + (Q_1(s)+Q(s))\log s\log^\varepsilon n\log\log n\log\log N\right),\end{aligned}$$

since a naive binary search gives $\vec{Q}_1(s) = O(Q_1(s) \log s)$ and $\vec{Q}(s) = O(Q(s) \log s)$. Then the structure in the final bootstrapping step that yielded (2) has

$$\vec{Q}(N, n) \quad = \quad O(\log_w s \log_s N + \log_w s \log s \log^{\varepsilon} n \log\log n \log\log N).$$

As $s = 2^{w^{1/2-\varepsilon}}$, we obtain $\vec{Q}(N, n) = O\left(\log_w N + w^{1-\Omega(\varepsilon)} \log\log N\right)$. As $w = \delta \log n$, we obtain the theorem. ◀

8 Future work

A number of interesting directions remain to be explored:

1. It would be nice to eliminate the assumption of polynomially bounded x-universe (i.e., the dependency in N). Our earlier paper [12] have already provided a mechanism to deal with a dynamic x-universe for the micro-structures in Lemma 4, but currently we have difficulty maintaining balance in the segment tree in Lemma 6 (standard weight-balanced B-trees seems to give an extra $\log_s n$ in the $U(s, n)$ term of the update time bound).
2. It would be nice to avoid randomization in our vertical ray shooting algorithm.
3. We have ignored space complexity throughout the paper. Many of the previous point location data structures achieves linear space. A naive upper bound on space for our data structure is n times the update time, i.e., $O\left(n \log^{1/2+\varepsilon} n\right)$. We believe that the bound can be lower by using more bit packing techniques, although it is unclear how to obtain linear space with our approach.
4. Insertion-only and deletion-only special cases are worth exploring. Here, $O\left(\frac{\log n}{\log\log n}\right)$ query time is not necessarily optimal; for example, see Wilkinson's insertion-only results on 2-d 3-sided orthogonal range searching [29]. As mentioned in the introduction, the deletion-only case has applications to geometric shortest paths [11].

References

1 Pankaj K. Agarwal, Lars Arge, Haim Kaplan, Eyal Molad, Robert E. Tarjan, and Ke Yi. An optimal dynamic data structure for stabbing-semigroup queries. *SIAM Journal on Computing*, 41(1):104–127, 2012. Preliminary version in SODA 2005. doi:10.1137/10078791X.

2 Stephen Alstrup, Thore Husfeldt, and Theis Rauhe. Marked ancestor problems. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 534–543, 1998. doi:10.1109/SFCS.1998.743504.

3 Arne Andersson and Mikkel Thorup. Dynamic ordered sets with exponential search trees. *Journal of the ACM*, 54(3), 2007. doi:10.1145/1236457.1236460.

4 Lars Arge. The buffer tree: A technique for designing batched external data structures. *Algorithmica*, 37(1):1–24, 2003. doi:10.1007/s00453-003-1021-x.

5 Lars Arge and Jeffrey Scott Vitter. Optimal external memory interval management. *SIAM Journal on Computing*, 32(6):1488–1508, 2003. doi:10.1137/S009753970240481X.

6 Guy E. Blelloch. Space-efficient dynamic orthogonal point location, segment intersection, and range reporting. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 894–903, 2008. URL: http://dl.acm.org/citation.cfm?id=1347082.1347180.

7 Timothy M. Chan. Geometric applications of a randomized optimization technique. *Discrete & Computational Geometry*, 22(4):547–567, 1999. doi:10.1007/PL00009478.

8 Timothy M. Chan. Persistent predecessor search and orthogonal point location on the word RAM. *ACM Transactions on Algorithms*, 9(3):22:1–22:22, 2013. Preliminary version in SODA 2011. doi:10.1145/2483699.2483702.

9 Timothy M. Chan and Yakov Nekrich. Towards an optimal method for dynamic planar point location. In *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 390–409, 2015. doi:10.1109/FOCS.2015.31.

10 Timothy M. Chan and Mihai Pătraşcu. Counting inversions, offline orthogonal range counting, and related problems. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 161–173, 2010. doi:10.1137/1.9781611973075.15.

11 Timothy M. Chan and Dimitrios Skrepetos. All-pairs shortest paths in unit-disk graphs in slightly subquadratic time. In *Proceedings of the 27th International Symposium on Algorithms and Computation (ISAAC)*, pages 24:1–24:13, 2016. doi:10.4230/LIPIcs.ISAAC.2016.24.

12 Timothy M. Chan and Konstantinos Tsakalidis. Dynamic orthogonal range searching on the RAM, revisited. In *Proceedings of the 33rd International Symposium on Computational Geometry (SoCG)*, pages 28:1–28:13, 2017. doi:10.4230/LIPIcs.SoCG.2017.28.

13 Richard Cole and Ramesh Hariharan. Dynamic LCA queries on trees. *SIAM Journal on Computing*, 34(4):894–923, 2005. doi:10.1137/S0097539700370539.

14 Paul F. Dietz. Fully persistent arrays. In *Proceedings of the 1st Workshop for Algorithms and Data Structures (WADS)*, pages 67–74, 1989. doi:10.1007/3-540-51542-9_8.

15 Michael Fredman and Michael Saks. The cell probe complexity of dynamic data structures. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC)*, pages 345–354, 1989. doi:10.1145/73007.73040.

16 Yoav Giyora and Haim Kaplan. Optimal dynamic vertical ray shooting in rectilinear planar subdivisions. *ACM Transactions on Algorithms*, 5(3):28:1–28:51, 2009. Preliminary version in SODA 2007. doi:10.1145/1541885.1541889.

17 Katherine Jane Lai. Complexity of union-split-find problems. Master's thesis, MIT, 2008. URL: http://hdl.handle.net/1721.1/45638.

18 Kurt Mehlhorn and Stefan Näher. Dynamic fractional cascading. *Algorithmica*, 5(1):215–241, 1990. doi:10.1007/BF01840386.

19 Christian Worm Mortensen. Fully-dynamic two dimensional orthogonal range and line segment intersection reporting in logarithmic time. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 618–627, 2003. URL: http://dl.acm.org/citation.cfm?id=644108.644210.

20 Christian Worm Mortensen. *Data structures for orthogonal intersection searching and other problems*. PhD thesis, IT University of Copenhagen, 2006. URL: http://www.epust.dk/main.pdf?attredirects=0.

21 Christian Worm Mortensen. Fully dynamic orthogonal range reporting on RAM. *SIAM Journal on Computing*, 35(6):1494–1525, 2006. doi:10.1137/S0097539703436722.

22 Christian Worm Mortensen, Rasmus Pagh, and Mihai Pătraşcu. On dynamic range reporting in one dimension. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, pages 104–111, 2005. doi:10.1145/1060590.1060606.

23 Yakov Nekrich. Searching in dynamic catalogs on a tree. *CoRR*, abs/1007.3415, 2010. arXiv:1007.3415.

24 Yakov Nekrich. A dynamic stabbing-max data structure with sub-logarithmic query time. In *Proceedings of the 22nd International Symposium on Algorithms and Computation (ISAAC)*, pages 170–179, 2011. doi:10.1007/978-3-642-25591-5_19.

25 Mark H. Overmars. *The Design of Dynamic Data Structures*, volume 156 of *Lecture Notes in Computer Science*. Springer, 1983. doi:10.1007/BFb0014927.

26 Yufei Tao. Dynamic ray stabbing. *ACM Transactions on Algorithms*, 11(2):11:1–11:19, 2014. doi:10.1145/2559153.

27 Peter van Emde Boas. Preserving order in a forest in less than logarithmic time and linear space. *Information Processing Letters*, 6(3):80–82, 1977. doi:10.1016/0020-0190(77)90031-X.

28 Peter van Emde Boas, Robert Kaas, and Erik Zijlstra. Design and implementation of an efficient priority queue. *Mathematical Systems Theory*, 10(1):99–127, 1976.

29 Bryan T. Wilkinson. Amortized bounds for dynamic orthogonal range reporting. In *Proceedings of the 22nd Annual European Symposium on Algorithms (ESA)*, pages 842–856, 2014. doi:10.1007/978-3-662-44777-2_69.

The Density of Expected Persistence Diagrams and its Kernel Based Estimation

Frédéric Chazal
Inria Saclay
Palaiseau, France
frederic.chazal@inria.fr

Vincent Divol
École Normale Supérieure
Paris, France
vincent.divol@ens.fr

Abstract

Persistence diagrams play a fundamental role in Topological Data Analysis where they are used as topological descriptors of filtrations built on top of data. They consist in discrete multisets of points in the plane $\mathbb{R}^2$ that can equivalently be seen as discrete measures in $\mathbb{R}^2$. When the data come as a random point cloud, these discrete measures become random measures whose expectation is studied in this paper. First, we show that for a wide class of filtrations, including the Čech and Rips-Vietoris filtrations, the expected persistence diagram, that is a deterministic measure on $\mathbb{R}^2$, has a density with respect to the Lebesgue measure. Second, building on the previous result we show that the persistence surface recently introduced in [1] can be seen as a kernel estimator of this density. We propose a cross-validation scheme for selecting an optimal bandwidth, which is proven to be a consistent procedure to estimate the density.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases topological data analysis, persistence diagrams, subanalytic geometry

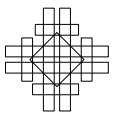
Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.26

Related Version A full version of the paper is available at `http://hal.archives-ouvertes.fr/hal-01716181` or `http://arxiv.org/abs/1802.10457`

Funding This work was partially supported by the Advanced Grant of the European Research Council GUDHI (Geometric Understanding in Higher Dimensions) and a collaborative research agreement between Inria and Fujitsu.

1 Introduction

Persistent homology [17], a popular approach in Topological Data Analysis (TDA), provides efficient mathematical and algorithmic tools to understand the topology of a point cloud by tracking the evolution of its homology at different scales. Specifically, given a scale (or time) parameter r and a point cloud $x = (x_1, \ldots, x_n)$ of size n, a simplicial complex $\mathcal{K}(x, r)$ is built on $\{1, \ldots, n\}$ thanks to some procedure, such as, e.g., the nerve of the union of balls of radius r centered on the point cloud or the Vietoris-Rips complex. Letting the scale r increase gives rise to an increasing sequence of simplicial complexes $\mathcal{K}(x) = (\mathcal{K}(x, r))_r$ called a *filtration*. When a simplex is added in the filtration at a time r, it either "creates" or "fills" some hole in the complex. Persistent homology keeps track of the birth and death of these holes and encodes them as a *persistence diagram* that can be seen as a relevant and stable [6, 7] multi-scale topological descriptor of the data. A persistence diagram D_s is thus a



collection of pairs of numbers, each of those pairs corresponding to the birth time and the death time of a s-dimensional hole. A precise definition of persistence diagram can be found, for example, in [17, 8]. Mathematically, a diagram is a multiset of points in

$$\Delta = \{\mathbf{r} = (r_1, r_2), 0 \leq r_1 < r_2 \leq \infty\}. \tag{1}$$

Note that in a general setting, points $\mathbf{r} = (r_1, r_2)$ in diagrams can be "at infinity" on the line $\{r_2 = \infty\}$ (e.g. a hole may never disappear). However, in the cases considered in this paper, this will be the case for a single point for 0-dimensional homology, and this point will simply be discarded in the following.

In statistical settings, one is often given a (i.i.d.) sample of (random) point clouds $\mathbb{X}_1, \ldots, \mathbb{X}_N$ and filtrations $\mathcal{K}(\mathbb{X}_1), \ldots, \mathcal{K}(\mathbb{X}_N)$ built on top of them. We consider the set of persistence diagrams $D_s[\mathcal{K}(\mathbb{X}_1)], \ldots, D_s[\mathcal{K}(\mathbb{X}_N)]$, which are thought to contain relevant topological information about the geometry of the underlying phenomenon generating the point clouds. The space of persistence diagrams is naturally endowed with the so-called *bottleneck distance* [13] or some variants. However, the resulting metric space turns out to be highly non linear, making the statistical analysis of distributions of persistence diagrams rather awkward, despite several interesting results such as, e.g., [27, 16, 11]. A common scheme to overcome this difficulty is to create easier to handle statistics by mapping the diagrams to a vector space thanks to a feature map Ψ, also called a representation (see, e.g., [1, 2, 4, 10, 12, 20, 24]). A classical idea to get information about the typical shape of a random point cloud is then to estimate the expectation $E[\Psi(D_s[\mathcal{K}(\mathbb{X}_i)])]$ of the distribution of representations using the mean representation

$$\overline{\Psi}_N := \frac{\sum_{i=1}^N \Psi(D_s[\mathcal{K}(\mathbb{X}_i)])}{N}. \tag{2}$$

In this direction, [4] introduces a representation called persistence landscape, and shows that it satisfies law of large numbers and central limit theorems. Similar theorems can be shown for a wide variety of representations: it is known that $\overline{\Psi}_N$ is a consistent estimator of $E[\Psi(D_s[\mathcal{K}(\mathbb{X}_i)])]$. Although it may be useful for a classification task, this mean representation is still somewhat disappointing from a theoretical point of view. Indeed, what exactly $E[\Psi(D_s[\mathcal{K}(\mathbb{X}_i)])]$ is, has been scarcely studied in a non-asymptotic setting, i.e. when the cardinality of the random point cloud $\mathbb{X}_i$ is fixed or bounded.

Asymptotic results, when the size of the considered point clouds goes to infinity, are well understood for some non-persistent descriptors of the data, such as the Betti numbers: a natural question in geometric probability is to study the asymptotics of the s-dimensional Betti numbers $\beta_s(\mathcal{K}(\mathbb{X}_n, r_n))$ where $\mathbb{X}_n$ is a point cloud of size n and under different asymptotics for r_n. Notable results on the topic include [18, 29, 30]. Considerably less results are known about the asymptotic properties of fundamentally persistent descriptors of the data: [3] finds the right order of magnitude of maximally persistent cycles and [15] shows the convergence of persistence diagrams on stationary process in a weak sense.

Contributions of the paper

In this paper, representing persistence diagrams as discrete measures, i.e. as element of the space of measures on $\mathbb{R}^2$, we establish non-asymptotic global properties of various representations and persistence-based descriptors. A multiset of points is naturally in bijection with the discrete measure defined on $\mathbb{R}^2$ created by putting Dirac measures on each point of the multiset, with mass equal to the multiplicity of the point. In this paper

a persistence diagram D_s is thus represented as a discrete measure on Δ and with a slight abuse of notation, we will write

$$D_s = \sum_{\mathbf{r} \in D_s} \delta_{\mathbf{r}}, \tag{3}$$

where $\delta_{\mathbf{r}}$ denotes the Dirac measure in $\mathbf{r}$ and where, as mentioned above, points with infinite persistence are simply discarded. A wide class of representations, including the persistence surface [1] (variants of this object have been also introduced [12, 20, 24]), the accumulated persistence function [2] or persistence silhouette [10] are conveniently expressed as $\Psi(D_s) = D_s(f) := \sum_{\mathbf{r} \in D_s} f(\mathbf{r})$ for some function f on Δ. Given a random set of points $\mathbb{X}$, the expected behavior of the representations $E[D_s[\mathcal{K}(\mathbb{X})](f)]$ is well understood if the expectation $E[D_s[\mathcal{K}(\mathbb{X})]]$ of the distribution of persistence diagrams is understood, where the expectation $E[\mu]$ of a random discrete measure μ is defined by the equation $E[\mu](B) = E[\mu(B)]$ for all Borel sets B (see [22] for a precise definition of $E[\mu]$ in a more general setting). Our main contribution (Theorem 7) consists in showing that for a large class of situations the expected persistence diagram $E[D_s[\mathcal{K}(\mathbb{X})]]$, which is a measure on $\Delta \subset \mathbb{R}^2$, has a density p with respect to the Lebesgue measure on $\mathbb{R}^2$. Therefore, $E[\Psi(D_s[\mathcal{K}(\mathbb{X})])]$ is equal to $\int pf$, and if properties of the density p are shown (such as smoothness), those properties will also apply to the expectation of the representation Ψ.

The main argument of the proof of Theorem 7 relies on the basic observation that for point clouds $\mathbb{X}$ of given size n, the filtration $\mathcal{K}(\mathbb{X})$ can induce a finite number of ordering configurations of the simplices. The core of the proof consists in showing that, under suitable assumptions, this ordering is locally constant for almost all $\mathbb{X}$. As one needs to use geometric arguments, having properties only satisfied almost everywhere is not sufficient for our purpose. One needs to show that properties hold in a stronger sense, namely that the set on which it is satisfied is a dense open set. Hence, a convenient framework to obtain such properties is given by subanalytic geometry [25]. Subanalytic sets are a class of subsets of $\mathbb{R}^d$ that are locally defined as linear projections of sets defined by analytic equations and inequations. As most considered filtrations in Topological Data Analysis result from real algebraic constructions, such sets naturally appear in practice. On open sets where the combinatorial structure of the filtration is constant, the way the points in the diagrams are matched to pairs of simplices is fixed: only the times/scales at which those simplices appear change. Under an assumption of smoothness of those times, and using the coarea formula [23], a classical result of geometric measure theory generalizing the change of variables formula in integrals, one then deduces the existence of a density for $E[D_s[\mathcal{K}(\mathbb{X})]]$.

Among the different representations of the form $\Psi(D) = D(f)$, persistence surface is of particular interest. It is defined as the convolution of a diagram with a gaussian kernel. Hence, the mean persistence surface can be seen as a kernel density estimator of the density p of Theorem 7. As a consequence, the general theory of kernel density estimation applies and gives theoretical guarantees about various statistical procedures. As an illustration, we consider the bandwidth selection problem for persistence surfaces. Whereas Adams et al. [1] states that any reasonable bandwidth is sufficient for a classification task, we give arguments for the opposite when no "obvious" shapes appear in the diagrams. We then propose a cross-validation scheme to select the bandwidth matrix. The consistency of the procedure is shown using Stone's theorem [26]. This procedure is implemented on a set of toy examples illustrating its relevance.

The paper is organized as follow: section 2 is dedicated to the necessary background in geometric measure theory and subanalytic geometry. Results are stated in section 3, and the main theorem is proved in section 4. It is shown in section 5 that the main result applies

to the Čech and Rips-Vietoris filtrations. Section 6 is dedicated to the statistical study of persistence surface, and numerical illustrations are found in section 7. All the technical proofs that are not essential to the understanding of the idea and results of the paper can be found in [9].

2 Preliminaries

2.1 The coarea formula

The proof of the existence of the density of the expected persistence diagram depends heavily on a classical result in geometric measure theory, the so-called coarea formula (see [23] for a gentle introduction to the subject). It consists in a more general version of the change of variables formula in integrals. Let (M, ρ) be a metric space. The diameter of a set $A \subset (M, \rho)$ is defined by $\sup_{x,y\in A} \rho(x, y)$.

▶ **Definition 1.** Let k be a non-negative integer. For $A \subset M$, and $\delta > 0$, consider

$$\mathcal{H}_k^\delta(A) := \inf\left\{\sum_i \operatorname{diam}(U_i)^k, A \subset \bigcup_i U_i \text{ and } \operatorname{diam}(U_i) < \delta\right\}. \tag{4}$$

The *k-dimensional Haussdorf measure* on M of A is defined by $\mathcal{H}_k(A) := \lim_{\delta\to 0} \mathcal{H}_k^\delta(A)$.

If M is a d-dimensional submanifold of $\mathbb{R}^D$, the d-dimensional Haussdorf measure coincides with the volume form associated to the ambient metric restricted to M. For instance, if M is an open set of $\mathbb{R}^D$, the Haussdorf measure is the D-dimensional Lebesgue measure.

▶ **Theorem 2** (Coarea formula [23])**.** *Let M (resp. N) be a smooth Riemannian manifold of dimension m (resp n). Assume that $m \geq n$ and let $\Phi : M \to N$ be a differentiable map. Denote by $D\Phi$ the differential of Φ. The Jacobian of Φ is defined by $J\Phi = \sqrt{\det((D\Phi) \times (D\Phi)^t)}$. For $f : M \to \mathbb{R}_+$ a positive measurable function, the following equality holds:*

$$\int_M f(x) J\Phi(x) d\mathcal{H}_m(x) = \int_N \left(\int_{x\in\Phi^{-1}(\{y\})} f(x) d\mathcal{H}_{m-n}(x) \right) d\mathcal{H}_n(y). \tag{5}$$

In particular, if $J\Phi > 0$ almost everywhere, one can apply the coarea formula to $f \times (J\Phi)^{-1}$ to compute $\int_M f$. Having $J\Phi > 0$ is equivalent to have $D\Phi$ of full rank: most of the proof of our main theorem consists in showing that this property holds for certain functions Φ of interest.

2.2 Background on subanalytic sets

We now give basic results on subanalytic geometry, whose proofs are given in [9]. See [25] for a thorough review of the subject. Let $M \subset \mathbb{R}^D$ be a connected real analytic submanifold possibly with boundary, whose dimension is denoted by d.

▶ **Definition 3.** A subset X of M is *semianalytic* if each point of M has a neighbourhood $U \subset M$ such that $X \cap U$ is of the form

$$\bigcup_{i=1}^{p} \bigcap_{j=1}^{q} X_{ij}, \tag{6}$$

where X_{ij} is either $f_{ij}^{-1}(\{0\})$ or $f_{ij}^{-1}((0, \infty))$ for some analytic functions $f_{ij} : U \to \mathbb{R}$.

▶ **Definition 4.** A subset X of M is *subanalytic* if for each point of M, there exists a neighborhood U of this point, a real analytic manifold N and A, a relatively compact semianalytic set of $N \times M$, such that $X \cap U$ is the projection of A on M. A function $f : X \to \mathbb{R}$ is subanalytic if its graph is subanalytic in $M \times \mathbb{R}$. The set of real-valued subanalytic functions on X is denoted by $\mathcal{S}(X)$.

A point x in a subanalytic subset X of M is smooth (of dimension k) if, in some neighbourhood of x in M, X is an analytic submanifold (of dimension k). The maximal dimension of a smooth point of X is called the dimension of X. The smooth points of X of dimension d are called regular, and the other points are called singular. The set $\mathrm{Reg}(X)$ of regular points of X is an open subset of M, possibly empty; the set of singular points is denoted by $\mathrm{Sing}(X)$.

▶ **Lemma 5.**

(i) *For $f \in \mathcal{S}(M)$, the set $A(f)$ on which f is analytic is an open subanalytic set of M. Its complement is a subanalytic set of dimension smaller than d.*

Fix X a subanalytic subset of M. Assume that $f, g : X \to \mathbb{R}$ are subanalytic functions such that the image of a bounded set is bounded. Then,

(ii) *The functions fg and $f + g$ are subanalytic.*

(iii) *The sets $f^{-1}(\{0\})$ and $f^{-1}((0, \infty))$ are subanalytic in M.*

As a consequence of point (i), for $f \in \mathcal{S}(M)$, one can define its gradient ∇f everywhere but on some subanalytic set of dimension smaller than d.

▶ **Lemma 6.** *Let X be a subanalytic subset of M. If the dimension of X is smaller than d, then $\mathcal{H}_d(X) = 0$.*

As a direct corollary, we always have

$$\mathcal{H}_d(X) = \mathcal{H}_d(\mathrm{Reg}(X)). \tag{7}$$

Write $\mathcal{N}(M)$ the class of subanalytic subsets X of M with $\mathrm{Reg}(X) = \emptyset$. We have just shown that $\mathcal{H}_d \equiv 0$ on $\mathcal{N}(M)$. They form a special class of negligeable sets. We say that a property is verified *almost subanalytically everywhere* (a.s.e.) if the set on which it is not verified is included in a set of $\mathcal{N}(M)$. For example, Lemma 5 implies that ∇f is defined a.s.e..

3 The density of expected persistence diagrams

Let $n > 0$ be an integer. Write $\mathcal{F}_n$ the collection of non-empty subsets of $\{1, \dots, n\}$. Let $\varphi = (\varphi[J])_{J \in \mathcal{F}_n} : M^n \to \mathbb{R}^{\mathcal{F}_n}$ be a continuous function. The function φ will be used to construct the persistence diagram and is called a *filtering function*: a simplex J is added in the filtration at the time $\varphi[J]$. Write for $x = (x_1, \dots, x_n) \in M^n$ and for J a simplex, $x(J) := (x_j)_{j \in J}$. We make the following assumptions on φ:

(K1) *Absence of interaction:* For $J \in \mathcal{F}_n$, $\varphi[J](x)$ only depends on $x(J)$.

(K2) *Invariance by permutation:* For $J \in \mathcal{F}_n$ and for $(x_1, \dots, x_n) \in M^n$, if τ is a permutation of $\{1, \dots, n\}$, then $\varphi[J](x_{\tau(1)}, \dots, x_{\tau(n)}) = \varphi[J](x_1, \dots, x_n)$.

(K3) *Monotony:* For $J \subset J' \in \mathcal{F}_n$, $\varphi[J] \leq \varphi[J']$.

(K4) *Compatibility:* For a simplex $J \in \mathcal{F}_n$ and for $j \in J$, if $\varphi[J](x_1, \dots, x_n)$ is not a function of x_j on some open set U of M^n, then $\varphi[J] \equiv \varphi[J \backslash \{j\}]$ on U.

(K5) *Smoothness:* The function φ is subanalytic and the gradient of each of its entries (which is defined a.s.e.) is non vanishing a.s.e..

Assumptions (K2) and (K3) ensure that a filtration $\mathcal{K}(x)$ can be defined thanks to φ by:

$$\forall J \in \mathcal{F}_n,\ J \in \mathcal{K}(x,r) \iff \varphi[J](x) \leq r. \tag{8}$$

Assumption (K1) means that the moment a simplex is added in the filtration only depends on the position of its vertices, but not on their relative position in the point cloud. For $J \in \mathcal{F}_n$, the gradient of $\varphi[J]$ is a vector field in TM^n. Its projection on the jth coordinate is denoted by $\nabla^j \varphi[J]$: it is a vector field in TM defined a.s.e.. The persistence diagram of the filtration $\mathcal{K}(x)$ for s-dimensional homology is denoted by $D_s[\mathcal{K}(x)]$.

▶ **Theorem 7.** *Fix $n \geq 1$. Assume that M is a real analytic compact d-dimensional connected submanifold possibly with boundary and that $\mathbb{X}$ is a random variable on M^n having a density with respect to the Haussdorf measure $\mathcal{H}_{dn}$. Assume that $\mathcal{K}$ satisfies the assumptions (K1)-(K5). Then, for $s \geq 0$, the expected measure $E[D_s[\mathcal{K}(\mathbb{X})]]$ has a density with respect to the Lebesgue measure on Δ.*

▶ Remark. The condition that M is compact can be relaxed in most cases: it is only used to ensure that the subanalytic functions appearing in the proof satisfy the boundedness condition of Lemma 5. For the Čech and Rips-Vietoris filtrations, one can directly verify that the function φ (and therefore the functions appearing in the proofs) satisfies it when $M = \mathbb{R}^d$. Indeed, in this case, the filtering functions are semi-algebraic.

Classical filtrations such as the Rips-Vietoris and Čech filtrations do not satisfy the full set of assumptions (K1)-(K5). Specifically, they do not satisfy the second part of assumption (K5): all singletons $\{j\}$ are included at time 0 in those filtrations so that $\varphi[\{j\}] \equiv 0$, and the gradient $\nabla\varphi[\{j\}]$ is therefore null everywhere. This leads to a well-known phenomenon on Rips-Vietoris and Čech diagrams: all the non-infinite points of the diagram for 0-dimensional homology are included in the vertical line $\{0\} \times [0, \infty)$. A theorem similar to Theorem 7 still holds in this case:

▶ **Theorem 8.** *Fix $n \geq 1$. Assume that M is a real analytic compact d-dimensional connected submanifold and that $\mathbb{X}$ is a random variable on M^n having a density with respect to the Haussdorf measure $\mathcal{H}_{dn}$. Define assumption (K5'):*

(K5') *The function φ is subanalytic and the gradient of its entries J of size greater than 1 is non vanishing a.s.e.. Moreover, for $\{j\}$ a singleton, $\varphi[\{j\}] \equiv 0$.*

Assume that $\mathcal{K}$ satisfies the assumptions (K1)-(K4) and (K5'). Then, for $s \geq 1$, $E[D_s[\mathcal{K}(\mathbb{X})]]$ has a density with respect to the Lebesgue measure on Δ. Moreover, $E[D_0[\mathcal{K}(\mathbb{X})]]$ has a density with respect to the Lebesgue measure on the vertical line $\{0\} \times [0, \infty)$.

The proof of Theorem 8 is very similar to the proof of Theorem 7. It is therefore relegated to the full version [9].

One can easily generalize Theorem 7 and assume that the size of the point process $\mathbb{X}$ is itself random. For $n \in \mathbb{N}$, define a function $\varphi^{(n)} : M^n \to \mathbb{R}^{\mathcal{F}_n}$ satisfying the assumption (K1)-(K5). If x is a finite subset of M, define $\mathcal{K}(x)$ by the filtration associated to $\varphi^{(|x|)}$ where $|x|$ is the size of x. We obtain the following corollary, proven in [9].

▶ **Corollary 9.** *Assume that $\mathbb{X}$ has some density with respect to the law of a Poisson process on M of intensity $\mathcal{H}_d$, such that $E\left[2^{|\mathbb{X}|}\right] < \infty$. Assume that $\mathcal{K}$ satisfies the assumptions (K1)-(K5). Then, for $s \geq 0$, $E[D_s[\mathcal{K}(\mathbb{X})]]$ has a density with respect to the Lebesgue measure on Δ.*

The condition $E\left[2^{|\mathbb{X}|}\right] < \infty$ ensures the existence of the expected diagram and is for example satisfied when $\mathbb{X}$ is a Poisson process with finite intensity.

As the way the filtration is created is smooth, one may actually wonder whether the density of $E[D_s[\mathcal{K}(\mathbb{X})]]$ is smooth as well: it is the case as long as the way the points are sampled is smooth. Recalling that a function is said to be of class C^k if it is k times differentiable, with a continuous kth derivative, we have the following result.

▶ **Theorem 10.** *Fix $0 \leq k \leq \infty$ and assume that $\mathbb{X} \in M^n$ has some density of class C^k with respect to $\mathcal{H}_{nd}$. Then, for $s \geq 0$, the density of $E[D_s[\mathcal{K}(\mathbb{X})]]$ is of class C^k.*

The proof is based on classical results of continuity under the integral sign as well as an use of the implicit function theorem: it can be found in [9].

As a corollary of Theorem 10, we obtain the smoothness of various expected descriptors computed on persistence diagrams. For instance, the expected birth distribution and the expected death distribution have smooth densities under the same hypothesis, as they are obtained by projection of the expected diagram on some axis. Another example is the smoothness of the expected Betti curves. The sth Betti number $\beta_s^r(\mathcal{K}(x))$ of a filtration $\mathcal{K}(x)$ is defined as the dimension of the sth homology group of $\mathcal{K}(x, r)$. The Betti curves $r \mapsto \beta_s^r(\mathcal{K}(x))$ are step functions which can be used as statistics, as in [28] where they are used for a classification task on time series. With few additional work (see proof in [9]), the expected Betti curves are shown to be smooth.

▶ **Corollary 11.** *Under the same hypothesis than Theorem 10, for $s \geq 0$, the expected Betti curve $r \mapsto E[\beta_s^r(\mathcal{K}(\mathbb{X}))]$ is a C^k function.*

4 Proof of Theorem 7

First, one can always replace M^n by $A(\varphi) = \bigcap_{J \in \mathcal{F}_n} A(\varphi[J])$, as Lemma 5 implies that it is an open set whose complement is in $\mathcal{N}(M^n)$. We will therefore assume that φ is analytic on M^n.

Given $x \in M^n$, the different values taken by $\varphi(x)$ on the filtration can be written $r_1 < \cdots < r_L$. Define $E_l(x)$ the set of simplices J such that $\varphi[J](x) = r_l$. The sets $E_1(x), \ldots, E_L(x)$ form a partition of $\mathcal{F}_n$ denoted by $\mathcal{A}(x)$.

▶ **Lemma 12.** *For a.s.e. $x \in M^n$, for $l \geq 1$, $E_l(x)$ has a minimal element J_l (for the partial order induced by inclusion).*

Proof. Fix $J, J' \subset \{1, \ldots, n\}$ with $J \neq J'$ and $J \cap J' \neq \emptyset$. consider the subanalytic functions $f : x \in M^n \mapsto \varphi[J](x) - \varphi[J'](x)$ and $g : x \in M^n \mapsto \varphi[J](x) - \varphi[J \cap J'](x)$. The set

$$C(J, J') := \{f = 0\} \cap \{g > 0\}. \tag{9}$$

is a subanalytic subset of M^n. Assume that it contains some open set U. On U, $\varphi[J](x)$ is equal to $\varphi[J'](x)$. Therefore, it does not depend on the entries x_j for $j \in J \backslash J'$. Hence, by assumption (K4), $\varphi[J](x)$ is actually equal to $\varphi[J \cap J'](x)$ on U. This is a contradiction with having $g > 0$ on U. Therefore, $C(J, J')$ does not contain any open set, and all its points are singular: $C(J, J')$ is in $\mathcal{N}(M^n)$. If $J \cap J' = \emptyset$, similar arguments show that $C(J, J') = \{f = 0\}$ cannot contain any open set: it would contradict assumption (K5). On the complement of

$$C := \bigcup_{J \neq J' \subset \{1, \ldots, n\}} C(J, J'), \tag{10}$$

having $\varphi[J](x) = \varphi[J'](x)$ implies that this quantity is equal to $\varphi[J \cap J'](x)$. This show the existence of a minimal element J_l to $E_l(x)$ on the complement of C. This property is therefore a.s.e. satisfied. ◀

▶ **Lemma 13.** *A.s.e., $x \mapsto \mathcal{A}(x)$ is locally constant.*

Proof. Fix $\mathcal{A}_0 = \{E_1, \dots, E_l\}$ a partition of $\mathcal{F}_n$ induced by some filtration, with minimal elements $J_1, \dots, J_l$. Consider the subanalytic functions F, G defined, for $x \in M^n$, by

$$F(x) = \sum_{l=1}^{L} \sum_{J \in E_l} (\varphi[J](x) - \varphi[J_l](x)) \text{ and } G(x) = \sum_{l \neq l'} (\varphi[J_l](x)) - \varphi[J_{l'}](x))^2 .$$

The set $\{x \in M^n, \mathcal{A}(x) = \mathcal{A}_0\}$ is exactly the set $C(\mathcal{A}_0) = \{F = 0\} \cap \{G > 0\}$, which is subanalytic. The sets $C(\mathcal{A}_0)$ for all partitions $\mathcal{A}_0$ of $\mathcal{F}_n$ define a finite partition of the space M^n. On each open set $\mathrm{Reg}(C(\mathcal{A}_0)))$, the application $x \mapsto \mathcal{A}(x)$ is constant. Therefore, $x \mapsto \mathcal{A}(x)$ is locally constant everywhere but on $\bigcup_{\mathcal{A}_0} \mathrm{Sing}(C(\mathcal{A}_0)) \in \mathcal{N}(M^n)$. ◀

Therefore, the space M^n is partitioned into a negligeable set of $\mathcal{N}(M^n)$ and some open subanalytic sets $U_1, \dots, U_R$ on which $\mathcal{A}$ is constant.

▶ **Lemma 14.** *Fix $1 \leq r \leq R$ and assume that $J_1, \dots, J_L$ are the minimal elements of $\mathcal{A}$ on U_r. Then, for $1 \leq l \leq L$ and $j \in J_l$, $\nabla^j \varphi[J_l] \neq 0$ a.s.e. on U_r.*

Proof. By minimality of J_l, for $j \in J_l$, the subanalytic set $\{\nabla^j \varphi[J_l] = 0\} \cap U_r$ cannot contain an open set. It is therefore in $\mathcal{N}(M^n)$. ◀

Fix $1 \leq r \leq R$ and write

$$V_r = U_r \setminus \left(\bigcup_{l=1}^{L} \bigcup_{j=1}^{|J_l|} \{\nabla^j \varphi[J_l] = 0\} \right).$$

The complement of V_r in U_r is still in $\mathcal{N}(M^n)$. For $x \in V_r$, $D_s[\mathcal{K}(x)]$ is written $\sum_{i=1}^{N} \delta_{\mathbf{r}_i}$, where $\mathbf{r}_i = (\varphi[J_{l_1}](x), \varphi[J_{l_2}](x)) =: (b_i, d_i)$. The integer N and the simplices J_{l_1}, J_{l_2} depend only on V_r. Note that d_i is always greater than b_i, so that J_{l_2} cannot be included in J_{l_1}. The map $x \mapsto \mathbf{r}_i$ has it differential of rank 2. Indeed, take $j \in J_{l_2} \backslash J_{l_1}$. By Lemma 14, $\nabla^j \varphi[J_{l_2}](x) \neq 0$. Also, as $\varphi[J_{l_1}]$ only depends on the entries of x indexed by J_{l_1} (assumption (K1)), $\nabla^j \varphi[J_{l_1}](x) = 0$. Furthermore, take j' in J_{l_1}. By Lemma 14, $\nabla^{j'} \varphi[J_{l_1}](x) \neq 0$. This implies that the differential is of rank 2.

We now compute the sth persistence diagram for $s \geq 0$. Write κ the density of $\mathbb{X}$ with respect to the measure $\mathcal{H}_{nd}$ on M^n. Then,

$$\begin{aligned} E[D_s[\mathcal{K}(\mathbb{X})]] &= \sum_{r=1}^{R} E\left[\mathbb{1}\{\mathbb{X} \in V_r\} D_s[\mathcal{K}(\mathbb{X})]\right] = \sum_{r=1}^{R} E\left[\mathbb{1}\{\mathbb{X} \in V_r\} \sum_{i=1}^{N_r} \delta_{\mathbf{r}_i}\right] \\ &= \sum_{r=1}^{R} \sum_{i=1}^{N_r} E\left[\mathbb{1}\{\mathbb{X} \in V_r\} \delta_{\mathbf{r}_i}\right] \end{aligned}$$

Write μ_{ir} the measure $E[\mathbb{1}\{\mathbb{X} \in V_r\}\delta_{\mathbf{r}_i}]$. To conclude, it suffices to show that this measure has a density with respect to the Lebesgue measure on Δ. This is a consequence of the coarea formula. Define the function $\Phi_{ir} : x \in V_r \mapsto \mathbf{r}_i = (\varphi[J_{l_1}](x), \varphi[J_{l_2}](x))$. We have already seen that Φ_{ir} is of rank 2 on V_r, so that $J\Phi_{ir} > 0$. By the coarea formula (see Lemma 2), for a Borel set B in Δ,

$$\begin{aligned} \mu_{ir}(B) = P(\Phi_{ir}(\mathbb{X}) \in B, \mathbb{X} \in V_r) &= \int_{V_r} \mathbb{1}\{\Phi_{ir}(x) \in B\} \kappa(x) d\mathcal{H}_{nd}(x) \\ &= \int_{u \in B} \int_{x \in \Phi_{ir}^{-1}(\{u\})} (J\Phi_{ir}(x))^{-1} \kappa(x) d\mathcal{H}_{nd-2}(x) du. \end{aligned}$$

Therefore, μ_{ir} has a density with respect to the Lebesgue measure on Δ equal to

$$p_{ir}(u) = \int_{x \in \Phi_{ir}^{-1}(\{u\})} (J\Phi_{ir}(x))^{-1} \kappa(x) d\mathcal{H}_{nd-2}(x). \tag{11}$$

Finally, $E[D_s[\mathcal{K}(\mathbb{X})]]$ has a density equal to

$$p(u) = \sum_{r=1}^{R} \sum_{i=1}^{N_r} \int_{x \in \Phi_{ir}^{-1}(\{u\})} (J\Phi_{ir}(x))^{-1} \kappa(x) d\mathcal{H}_{nd-2}(x). \tag{12}$$

▶ Remark. Notice that, for n fixed, the above proof, and thus the conclusion, of Theorem 7 also works if the diagrams are represented by normalized discrete measures, i.e. probability measures defined by

$$D_s = \frac{1}{|D_s|} \sum_{\mathbf{r} \in D_s} \delta_{\mathbf{r}}. \tag{13}$$

5 Examples

We now note that the Rips-Vietoris and the Čech filtrations satisfy the assumptions (K1)-(K4) and (K5') when $M = \mathbb{R}^d$ is an Euclidean space. Note that the similar arguments show that weighted versions of those filtrations (see [5]) satisfy assumptions (K1)-(K5).

5.1 Rips-Vietoris filtration

For the Rips-Vietoris filtration, $\varphi[J](x) = \max_{i,j \in J} \|x_i - x_j\|$. The function φ clearly satisfies (K1), (K2) and (K3). It is also subanalytic, as it is the maximum of semi-algebraic functions.

For $x \in M^n$ and $J \in \mathcal{F}_n$ a simplex of size greater than one, $\varphi[J](x) = \|x_i - x_j\|$ for some indices i, j. Those indices are locally stable, and $\varphi[J](x) = \varphi[\{i,j\}](x)$: hypothesis (K4) is satisfied. Furthermore, on this set,

$$\nabla \varphi[\{i,j\}](x) = \left(\frac{x_i - x_j}{\|x_i - x_j\|}, \frac{x_j - x_i}{\|x_i - x_j\|} \right) \neq 0. \tag{14}$$

Hence, (K5') is also satisfied: both Theorem 8 and Theorem 10 are satisfied for the Rips-Vietoris filtration.

5.2 Čech filtration

The ball centered at x of radius r is denoted by $B(x,r)$. For the Čech filtration,

$$\varphi[J](x) = \inf_{r>0} \left\{ \bigcap_{j \in J} B(x_j, r) \neq \emptyset \right\}. \tag{15}$$

First, it is clear that (K1), (K2) and (K3) are satisfied by φ.

We give without proof a characterization of the Čech complex.

▶ **Proposition 15.** *Let x be in M^n and fix $J \in \mathcal{F}_n$. If the circumcenter of $x(J)$ is in the convex hull of $x(J)$, then $\varphi[J](x)$ is the radius of the circumsphere of $x(J)$. Otherwise, its projection on the convex hull belongs to the convex hull of some subsimplex $x(J')$ of $x(J)$ and $\varphi[J](x) = \varphi[J'](x)$.*

▶ **Definition 16.** The Cayley-Menger matrix of a k-simplex $x = (x_1, \dots, x_k) \in M^k$ is the symmetric matrix $(M(x)_{i,j})_{i,j}$ of size $k+1$, with zeros on the diagonal, such that $M(x)_{1,j} = 1$ for $j > 1$ and $M(x)_{i+1,j+1} = \|x_i - x_j\|^2$ for $i, j \leq k$.

▶ **Proposition 17** (see [14]). *Let $x \in M^k$ be a point in general position. Then, the Cayley-Menger matrix $M(x)$ is invertible with $(M(x))^{-1}_{1,1} = -2r^2$, where r is the radius of the circumsphere of x. The kth other entries of the first line of $M(x)^{-1}$ are the barycentric coordinates of the circumcenter.*

Therefore, the application which maps a simplex to its circumcenter is analytic, and the set on which the circumcenter of a simplex belongs in the interior of its convex hull is a subanalytic set. On such a set, the function φ is also analytic, as it is the square root of the inverse a matrix which is polynomial in x. Furthermore, on the open set on which the circumcenter is outside the convex hull, we have shown that $\varphi[J](x) = \varphi[J'](x)$ for some subsimplex J': assumption (K4) is satisfied.

Finally, let us show that assumption (K5') is satisfied. The previous paragraph shows the subanalyticity of φ. For $J \in \mathcal{F}_n$ a simplex of size greater than one, there exists some subsimplex J' such that $\varphi[J](x)$ is the radius of the circumsphere of $x(J')$. It is clear that there cannot be an open set on which this radius is constant. Thus, $\nabla\varphi[J]$ is a.s.e. non null.

6 Persistence surface as a kernel density estimator

Persistence surface is a representation of persistence diagrams introduced by Adams & al. in [1]. It consists in a convolution of a diagram with a kernel, a general idea that has been repeatedly and fruitfully exploited, with slight variations, for instance in [12, 20, 24]. For $K : \mathbb{R}^2 \to \mathbb{R}$ a kernel and H a bandwidth matrix (e.g. a symmetric positive definite matrix), let for $u \in \mathbb{R}^2$,

$$K_H(u) = \det(H)^{-1/2} K(H^{-1/2} \cdot u). \tag{16}$$

For D a diagram, $K : \mathbb{R}^2 \to \mathbb{R}$ a kernel, H a bandwidth matrix and $w : \mathbb{R}^2 \to \mathbb{R}_+$ a weight function, one defines the persistence surface of D with kernel K and weight function w by:

$$\forall u \in \mathbb{R}^2,\ \rho(D)(u) := \sum_{\mathbf{r} \in D} w(\mathbf{r}) K_H(u - \mathbf{r}) = D(wK_H(u - \cdot)) \tag{17}$$

Assume that $\mathbb{X}$ is some point process satisfying the assumptions of Theorem 7. Then, for $s \geq 1$, $\mu := E[D_s[\mathcal{K}(\mathbb{X})]]$ has some density p with respect to the Lebesgue measure on Δ. Therefore, μ_w, the measure having density w with respect to μ, has a density equal to $w \times p$ with respect to the Lebesgue measure. The mean persistence surface $E[\rho(D_s[\mathcal{K}(\mathbb{X})])]$ is exactly the convolution of μ_w by some kernel function: the persistence surface $\rho(D_s[\mathcal{K}(\mathbb{X})])$ is actually a kernel density estimator of $w \times p$.

If a point cloud approximates a shape, then its persistence diagram (for the Čech filtration for instance) is made of numerous points with small persistences and a few meaningful points of high persistences which corresponds to the persistence diagram of the "true" shape. As one is interested in the latter points, a weight function w, which is typically an increasing function of the persistence, is used to suppress the importance of the topological noise in the persistence surface. Adams & al. [1] argue that in this setting, the choice of the bandwidth matrix H has few effects for statistical purposes (e.g. classification), a claim supported by numerical experiments on simple sets of synthetic data, e.g. torus, sphere, three clusters, etc.

However, in the setting where the datasets are more complicated and contain no obvious "real" shapes, one may expect the choice of the bandwidth parameter H to become more critical: there are no highly persistent, easily distinguishable points in the diagrams anymore and the precise structure of the density functions of the processes becomes of interest. We now show that a cross validation approach allows the bandwidth selection task to be done in an asymptotically consistent way. This is a consequence of a generalization of Stone's theorem [26] when observations are not random vectors but random measures.

Assume that $\mu_1, \dots, \mu_N$ are i.i.d. random measures on $\mathbb{R}^2$, such that there exists a deterministic constant C with $|\mu_1| \leq C$. Assume that the expected measure $E[\mu_1]$ has a bounded density p with respect to the Lebesgue measure on $\mathbb{R}^2$. Given a kernel $K : \mathbb{R}^2 \to \mathbb{R}$ and a bandwidth matrix H, one defines the kernel density estimator

$$\hat{p}_H(x) := \frac{1}{N} \sum_{i=1}^{N} \int K_H(x-y)\mu_i(dy). \tag{18}$$

The optimal bandwidth H_{opt} minimizes the Mean Integrated Square Error (MISE)

$$MISE(H) := E\left[\|p - \hat{p}_H\|^2\right] = E\left[\int (p(x) - \hat{p}_H(x))^2 \, dx\right]. \tag{19}$$

Of course, as p is unknown, $MISE(H)$ cannot be computed. Minimizing $MISE(H)$ is equivalent to minimize $J(H) := MISE(H) - \|p\|^2$. Define

$$\hat{p}_{iH}(x) := \frac{1}{N-1} \sum_{j \neq i} \int K_H(x-y)\mu_j(dy) \tag{20}$$

and

$$\hat{J}(H) := \frac{1}{N^2} \sum_{i,j} \iint K_H^{(2)}(x-y)\mu_i(dx)\mu_j(dy) - \frac{2}{N} \sum_i \int \hat{p}_{iH}(x)\mu_i(dx), \tag{21}$$

where $K^{(2)} : x \mapsto \int K(x-y)K(y)dy$ denotes the convolution of K with itself. The quantity $\hat{J}(H)$ is an unbiased estimator of $J(H)$. The selected bandwidth $\hat{H}$ is then chosen to be equal to $\arg\min_H \hat{J}(H)$.

▶ **Theorem 18** (Stone's theorem [26]). *Assume that the kernel K is nonnegative, Hölder continuous and has a maximum attained in 0. Also assume that the density p is bounded. Then, $\hat{H}$ is asymptotically optimal in the sense that*

$$\frac{\|p - \hat{p}_{\hat{H}}\|}{\|p - \hat{p}_{H_{opt}}\|} \xrightarrow[N\to\infty]{} 1 \ \ a.s.. \tag{22}$$

Note that the gaussian kernel $K(x) = \exp(-\|x\|^2/2)$ satisfies the assumptions of Theorem 18.

Let $\mathbb{X}_1, \dots, \mathbb{X}_N$ be i.i.d. processes on M having a density with respect to the law of a Poisson process of intensity $\mathcal{H}_d$. Assume that there exists a deterministic constant C with $|\mathbb{X}_i| \leq C$. Then, Theorem 18 can be applied to $\mu_i = D_s[\mathcal{K}(\mathbb{X}_i)]$. Therefore, *the cross validation procedure* (21) *to select H the bandwidth matrix in the persistence surface ensures that the mean persistence surface*

$$\overline{\rho}_N := \frac{1}{N} \sum_{i=1}^{N} \rho(D_s[\mathcal{K}(\mathbb{X}_i)]) \tag{23}$$

is a good estimator of p the density of $E[D_s[\mathcal{K}(\mathbb{X}_1)]]$.

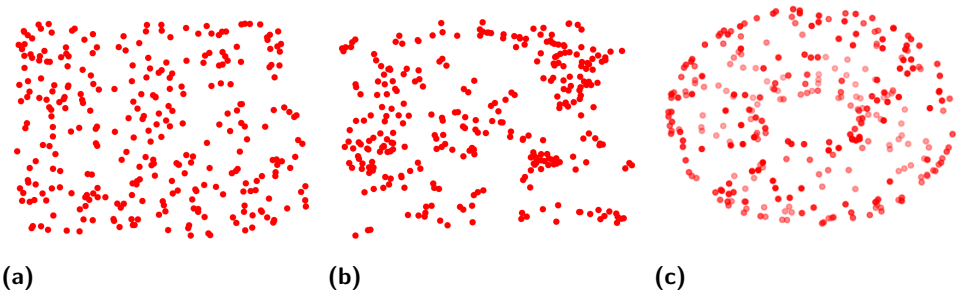


Figure 1 Realization of the processes (a), (b) and (c) described in Section 7.

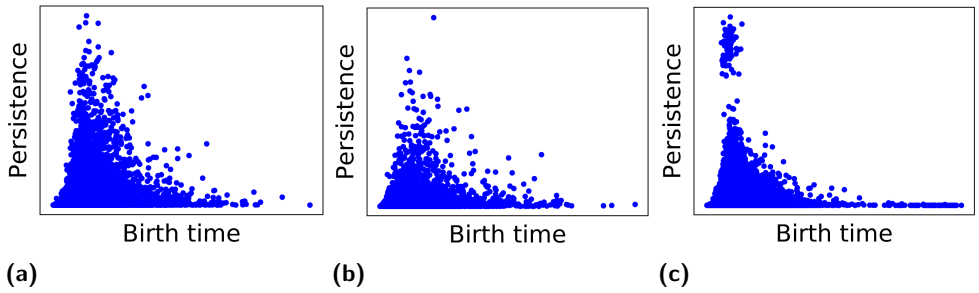
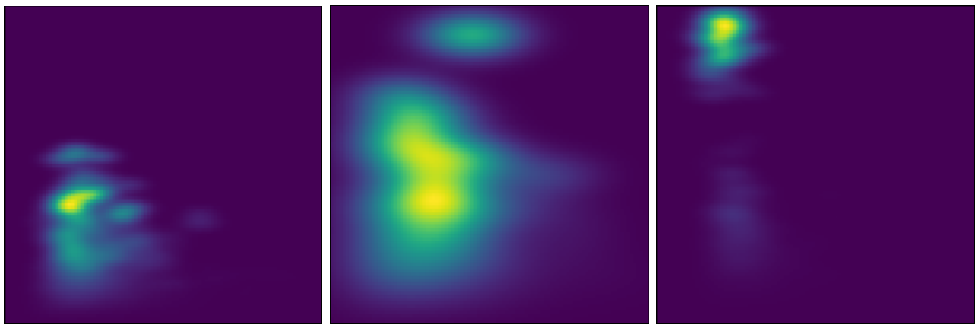


Figure 2 Superposition of the $N = 40$ diagrams of class (a), (b) and (c), transformed under the map $\mathbf{r} \to (r_1, r_2 - r_1)$.

7 Numerical illustration

Three sets of synthetic data are considered (see Figure 1). The first one (a) is made of $N = 40$ sets of $n = 300$ i.i.d. points uniformly sampled in the square $[0, 1]^2$. The second one (b) is made of N samples of a clustered process: $n/3$ cluster's centers are uniformly sampled in the square. Each center is then replaced with 3 i.i.d. points following a normal distribution of standard deviation $0.01 \times n^{-1/2}$. The third dataset (c) is made of N samples of n uniform points on a torus of inner radius 1 and outer radius 2. For each set, a Čech persistence diagram for 1-dimensional homology is computed. Persistence diagrams are then transformed under the map $(r_1, r_2) \mapsto (r_1, r_2 - r_1)$, so that they now live in the upper-left quadrant of the plane. Figure 2 shows the superposition of the diagrams in each class. One may observe the slight differences in the structure of the topological noise over the classes (a) and (b). The cluster of most persistent points in the diagrams of class (c) correspond to the two holes of a torus and are distinguishable from the rest of the points in the diagrams of the class, which form topological noise. The persistence diagrams are weighted by the weight function $w(\mathbf{r}) = (r_2 - r_1)^3$, as advised in [19] for two-dimensional point clouds. The bandwidth selection procedure will be applied to the measures having density w with respect to the diagrams, e.g. a measure is a sum of weighted Dirac measures.

For each class of dataset, the score $\hat{J}(H)$ is computed for a set of bandwidth matrices of the form $h^2 \times \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, for 50 values h evenly spaced on a log-scale between 10^{-5} and 1. Note that the computation of $\hat{J}(H)$ only involves the computations of $K_H(\mathbf{r}_1 - \mathbf{r}_2)$ for points $\mathbf{r}_1$, $\mathbf{r}_2$ in different diagrams. Hence, the complexity of the computation of $\hat{J}(H)$ is in $O(T^2)$, where T is the sum of the number of points in the diagrams of a given class. If this is too costly, one may use a subsampling approach to estimate the integrals. The



(a) (b) (c)

Figure 3 Persistence surfaces for each class (a), (b) and (c), computed with the weight function $w(\mathbf{r}) = (r_2 - r_1)^3$ and with the bandwidth matrix selected by the cross-validation procedure.

selected bandwidth were respectively $h = 0.22, 0.60, 0.17$. Persistence surfaces for the selected bandwidth are displayed in Figure 3. The persistence of the "true" points of the torus are sufficient to suppress the topological noise: only two yellow areas are seen in the persistence surface of the torus. Note that the two areas can be separated, whereas it is not obvious when looking at the superposition of the diagrams, and would not have been obvious with an arbitrary choice of bandwidth. The bandwidth for class (b) may look to have been chosen too big. However, there is much more variability in class (b) than in the other classes: this phenomenon explains that the density is less peaked around a few selected areas than in class (a).

Illustrations on non-synthetic data are shown in [9]: similar behaviors are observed.

8 Conclusion and further works

Taking a measure point of view to represent persistence diagrams, we have shown that the expected behavior of persistence diagrams built on top of random point sets reveals to have a simple and interesting structure: a measure on $\mathbb{R}^2$ with density with respect to Lebesgue measure that is as smooth as the random process generating the data points! This opens the door to the use of effective kernel density estimation techniques for the estimation of the expectation of topological features of data. Our approach and results also seem to be particularly well-suited to the use of recent results on the Lepski method for parameter selection [21] in statistics, a research direction that deserves further exploration. As many persistence-based features considered among the literature - persistence images, birth and death distributions, Betti curves,... - can be expressed as linear functional of the discrete measure representation of diagrams, our results immediately extend to them. The ability to select the parameters on which these features are dependent in a well-founded statistical way also opens the door to a well-justified usage of persistence-based features in further supervised and un-supervised learning tasks.

References

1 Henry Adams, Tegan Emerson, Michael Kirby, Rachel Neville, Chris Peterson, Patrick Shipman, Sofya Chepushtanova, Eric Hanson, Francis Motta, and Lori Ziegelmeier. Persistence images: a stable vector representation of persistent homology. *Journal of Machine Learning Research*, 18(8):1–35, 2017.

2 Christophe Biscio and Jesper Møller. The accumulated persistence function, a new useful functional summary statistic for topological data analysis, with a view to brain artery trees and spatial point process applications. *arXiv preprint arXiv:1611.00630*, 2016.

3 Omer Bobrowski, Matthew Kahle, Primoz Skraba, et al. Maximally persistent cycles in random geometric complexes. *The Annals of Applied Probability*, 27(4):2032–2060, 2017.

4 Peter Bubenik. Statistical topological data analysis using persistence landscapes. *The Journal of Machine Learning Research*, 16(1):77–102, 2015.

5 Mickaël Buchet, Frédéric Chazal, Steve Y Oudot, and Donald R Sheehy. Efficient and robust persistent homology for measures. *Computational Geometry*, 58:70–96, 2016.

6 F. Chazal, D. Cohen-Steiner, L. J. Guibas, F. Memoli, and S. Y. Oudot. Gromov-hausdorff stable signatures for shapes using persistence. *Computer Graphics Forum (proc. SGP 2009)*, pages 1393–1403, 2009.

7 F. Chazal, V. de Silva, and S. Oudot. Persistence stability for geometric complexes. *Geometriae Dedicata*, 173(1):193–214, 2014.

8 Frédéric Chazal, Vin de Silva, Marc Glisse, and Steve Oudot. *The structure and stability of persistence modules*. SpringerBriefs in Mathematics. Springer, 2016.

9 Frédéric Chazal and Vincent Divol. The density of expected persistence diagrams and its kernel based estimation. Extended version of a paper to appear in the proceedings of the Symposium of Computational Geometry 2018, 2018. URL: `https://hal.archives-ouvertes.fr/hal-01716181`.

10 Frédéric Chazal, Brittany Terese Fasy, Fabrizio Lecci, Alessandro Rinaldo, and Larry Wasserman. Stochastic convergence of persistence landscapes and silhouettes. In *Proceedings of the thirtieth annual symposium on Computational geometry*, page 474. ACM, 2014.

11 Frédéric Chazal, Marc Glisse, Catherine Labruère, and Bertrand Michel. Convergence rates for persistence diagram estimation in topological data analysis. *Journal of Machine Learning Research*, 16:3603–3635, 2015. URL: `http://jmlr.org/papers/v16/chazal15a.html`.

12 Yen-Chi Chen, Daren Wang, Alessandro Rinaldo, and Larry Wasserman. Statistical analysis of persistence intensity functions. *arXiv preprint arXiv:1510.02502*, 2015.

13 David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. *Discrete & Computational Geometry*, 37(1):103–120, 2007.

14 HSM Coxeter. The circumradius of the general simplex. *The Mathematical Gazette*, pages 229–231, 1930.

15 Trinh Khanh Duy, Yasuaki Hiraoka, and Tomoyuki Shirai. Limit theorems for persistence diagrams. *arXiv preprint arXiv:1612.08371*, 2016.

16 B. T. Fasy, F. Lecci, A. Rinaldo, L. Wasserman, S. Balakrishnan, A. Singh, et al. Confidence sets for persistence diagrams. *The Annals of Statistics*, 42(6):2301–2339, 2014.

17 D. Morozov H. Edelsbrunner. Persistent homology. In *Handbook of Discrete and Computational Geometry (3rd Ed - To appear)*. CRC Press (to appear), 2017.

18 Matthew Kahle, Elizabeth Meckes, et al. Limit the theorems for betti numbers of random simplicial complexes. *Homology, Homotopy and Applications*, 15(1):343–374, 2013.

19 Genki Kusano, Kenji Fukumizu, and Yasuaki Hiraoka. Kernel method for persistence diagrams via kernel embedding and weight factor. *arXiv preprint arXiv:1706.03472*, 2017.

20 Genki Kusano, Yasuaki Hiraoka, and Kenji Fukumizu. Persistence weighted gaussian kernel for topological data analysis. In *International Conference on Machine Learning*, pages 2004–2013, 2016.

21 Claire Lacour, Pascal Massart, and Vincent Rivoirard. Estimator selection: a new method with applications to kernel density estimation. *arXiv preprint arXiv:1607.05091*, 2016.

22 Michel Ledoux and Michel Talagrand. *Probability in Banach Spaces: isoperimetry and processes*. Springer Science & Business Media, 2013.

23 F. Morgan. *Geometric Measure Theory: A Beginner's Guide*. Elsevier Science, 2016.

24 Jan Reininghaus, Stefan Huber, Ulrich Bauer, and Roland Kwitt. A stable multi-scale kernel for topological machine learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4741–4748, 2015.

25 M. Shiota. *Geometry of Subanalytic and Semialgebraic Sets*. Progress in mathematics. Springer, 1997.

26 Charles J Stone. An asymptotically optimal window selection rule for kernel density estimates. *The Annals of Statistics*, pages 1285–1297, 1984.

27 Katharine Turner, Yuriy Mileyko, Sayan Mukherjee, and John Harer. Fréchet means for distributions of persistence diagrams. *Discrete & Computational Geometry*, 52(1):44–70, 2014.

28 Yuhei Umeda. Time series classification via topological data analysis. *Transactions of the Japanese Society for Artificial Intelligence*, 32(3):D–G72_1, 2017.

29 D Yogeshwaran, Robert J Adler, et al. On the topology of random complexes built over stationary point processes. *The Annals of Applied Probability*, 25(6):3338–3380, 2015.

30 D. Yogeshwaran, Eliran Subag, and Robert J. Adler. Random geometric complexes in the thermodynamic regime. *Probability Theory and Related Fields*, 167(1):107–142, Feb 2017. doi:10.1007/s00440-015-0678-9.

Embedding Graphs into Two-Dimensional Simplicial Complexes

Éric Colin de Verdière[1]
Université Paris-Est, LIGM, CNRS, ENPC, ESIEE Paris, UPEM, Marne-la-Vallée
France
eric.colindeverdiere@u-pem.fr

Thomas Magnard[2]
Université Paris-Est, LIGM, CNRS, ENPC, ESIEE Paris, UPEM, Marne-la-Vallée
France
thomas.magnard@u-pem.fr

Bojan Mohar[3]
Department of Mathematics, Simon Fraser University
Burnaby, Canada
mohar@sfu.ca
https://orcid.org/0000-0002-7408-6148

Abstract

We consider the problem of deciding whether an input graph G admits a topological embedding into a two-dimensional simplicial complex $\mathscr{C}$. This problem includes, among others, the embeddability problem of a graph on a surface and the topological crossing number of a graph, but is more general.

The problem is NP-complete when $\mathscr{C}$ is part of the input, and we give a polynomial-time algorithm if the complex $\mathscr{C}$ is fixed.

Our strategy is to reduce the problem to an embedding extension problem on a surface, which has the following form: Given a subgraph H' of a graph G', and an embedding of H' on a surface S, can that embedding be extended to an embedding of G' on S? Such problems can be solved, in turn, using a key component in Mohar's algorithm to decide the embeddability of a graph on a fixed surface (STOC 1996, SIAM J. Discr. Math. 1999).

2012 ACM Subject Classification Theory of computation → Computational geometry, Mathematics of computing → Graph algorithms, Mathematics of computing → Graphs and surfaces, Mathematics of computing → Algebraic topology

Keywords and phrases computational topology, embedding, simplicial complex, graph, surface

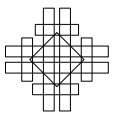
Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.27

Related Version A full version of this paper is available at `https://arxiv.org/abs/1803.07032`.

[1] Part of this work was done while at Département d'Informatique, École normale supérieure, Paris, France. This author is supported in part by grant ANR-17-CE40-0033 of the French National Research Agency ANR (SoS project).

[2] Part of this work was done while at Département d'Informatique, École normale supérieure, Paris, France. This author is supported in part by grant ANR-17-CE40-0033 of the French National Research Agency ANR (SoS project).

[3] Part of this work was done while this author was invited professor at Département d'Informatique, École normale supérieure, Paris, France. This author is supported in part by the NSERC Discovery Grant R611450 (Canada), by the Canada Research Chairs program, and by the Research Project J1-8130 of ARRS (Slovenia).



1 Introduction

Topological embedding problems. Topological embedding problems are among the most fundamental problems in computational topology, already emphasized since the early developments of this discipline [8, Section 10]. Their general form is as follows: Given topological spaces X and Y, does there exist an embedding (a continuous, injective map) from X to Y? Since a finite description of X and Y is needed, typically they are represented as finite simplicial complexes, which are topological spaces obtained by attaching simplices (points, segments, triangles, tetrahedra, etc.) of various dimensions tog1803.07032ether.

The case where the host space Y equals $\mathbb{R}^d$ (or, almost equivalently, $\mathbb{S}^d$, which can be modeled as a simplicial complex) has been studied the most. The case $d = 2$ corresponds to the planarity testing problem, which has attracted considerable interest [25]. The case $d = 3$ is much harder, and has only recently been shown to be decidable by Matoušek, Sedgwick, Tancer, and Wagner [19]. The general problem for arbitrary d has been extensively studied in the last few years, starting with hardness results by Matoušek, Tancer, and Wagner [20], and continuing with some algorithmic results in a series of articles; we refer to Matoušek et al. [19, Introduction] for a state of the art.

What about more general choices of Y? The case where Y is a graph is essentially the subgraph homeomorphism problem, asking if Y contains a subdivision of a graph X. This is hard in general, easy when Y is fixed, and polynomial-time solvable for every fixed X, by using graph minor algorithms. The case where X is a graph and Y a 2-dimensional simplicial complex that is homeomorphic to a surface has been much investigated, also in connection to topological graph theory [23] and algorithms for surface-embedded graphs [7, 11]: The problem is NP-complete, as proved by Thomassen [28], but Mohar [22] has proved that it can be solved in linear time if Y is fixed (in some recent works, the proof has been simplified and the result extended [13, 15]). The case where X is a 2-complex and Y is (a 2-complex homeomorphic to) a surface essentially boils down to the previous case; see Mohar [21]. More recently, Čadek, Krčál, Matoušek, Vokřínek, and Wagner [4, Theorem 1.4] considered the case where the host complex Y has an arbitrary (but fixed) dimension; they provide a polynomial-time algorithm for the related *map extension problem*, under some assumptions on the dimensions of X and Y; in particular, Y must have trivial fundamental group (because they manipulate in an essential way the homotopy groups of Y, which have to be Abelian); but the maps they consider need not be embeddings.

Another variation on this problem is to try to embed X such that it extends a given partial embedding of X (we shall consider such *embedding extension problems* later). This problem has already been studied in some particular cases; in particular, Angelini, Battista, Frati, Jelínek, Kratochvìl, Patrignani, and Rutter [1, Theorem 4.5] provide a linear-time algorithm to decide the embedding extension problem of a graph in the plane.

Our results. In this article, we study the topological embedding problem when X is an arbitrary graph G, and Y is an arbitrary two-dimensional simplicial complex $\mathscr{C}$ (actually, a simplicial complex of dimension at most two—abbreviated as *2-complex* below). Formally, we consider the following decision problem:

Embed(n, c):
Input: A graph G with n vertices and edges, and a 2-complex $\mathscr{C}$ with c simplices.
Question: Does G have a topological embedding into $\mathscr{C}$?

(We use the parameters n and c whenever we need to refer to the input size.) Here are our main results:

▶ **Theorem 1.** *The problem* Embed *is NP-complete.*

▶ **Theorem 2.** *The problem* Embed(n, c) *can be solved in time* $f(c) \cdot n^{O(c)}$*, where* f *is some computable function of* c*.*

As for Theorem 1, it is straightforward that the problem is NP-hard (as the case where $\mathscr{C}$ is a surface is already NP-hard); the interesting part is to provide a certificate checkable in polynomial time when an embedding exists. Note that Theorem 2 shows that, for every fixed complex $\mathscr{C}$, the problem of deciding whether an input graph embeds into $\mathscr{C}$ is polynomial-time solvable. Actually, our algorithm is explicit, in the sense that, if there exists an embedding of G on $\mathscr{C}$, we can provide some representation of such an embedding (in contrast to some results in the theory of graph minors, where the existence of an embedding can be obtained without leading to an explicit construction).

Why do 2-complexes look harder than surfaces? A key property of the class of graphs embeddable on a fixed surface is that it is minor-closed: Having a graph G embeddable on a surface $\mathscr{S}$, removing or contracting any edge yields a graph embeddable on $\mathscr{S}$. By Robertson and Seymour's theory, this immediately implies a cubic-time algorithm to test whether a graph G embeds on $\mathscr{S}$, for every fixed surface $\mathscr{S}$ [26]. In contrast, the class of graphs embeddable on a fixed 2-complex is, in general, not closed under taking minors, and thus this theory does not apply. For example, let $\mathscr{C}$ be obtained from two tori by connecting them together with a line segment, and let G be obtained from two copies of K_5 by joining them together with a new edge e; then G embeds into $\mathscr{C}$, but the minor obtained from G by contracting e does not.

Two-dimensional simplicial complexes are topologically much more complicated than surfaces. For example, there exist linear-time algorithms to decide whether two surfaces are homeomorphic (this amounts to comparing the Euler characteristics, the orientability characters, and, in case of surfaces with boundary, the numbers of boundary components), or to decide whether a closed curve is contractible (see Dey and Guha [9], Lazarus and Rivaud [16], and Erickson and Whittlesey [12]). In contrast, the homeomorphism problem for 2-complexes is as hard as graph isomorphism, as shown by Ó Dúnlaing, Watt, and Wilkins [24]. Moreover, the contractibility problem for closed curves on 2-complexes is undecidable; even worse, there exists a fixed 2-complex $\mathscr{C}$ such that the contractibility problem for closed curves on $\mathscr{C}$ is undecidable (this is because every finitely presented group can be realized as the fundamental group of a 2-complex, and there is such a group in which the word problem is undecidable, by a result of Boone [3]; see also Stillwell [27, Section 9.3]).

Despite this stark contrast between surfaces and 2-complexes, if we care only on the polynomiality or non-polynomiality, our results show that the complexities of embedding a graph into a surface or a 2-complex are similar: If the host space is not fixed, the problem is NP-complete; otherwise, it is polynomial-time solvable. Compared to the aforementioned hard problems on general 2-complexes, one feature related to our result is that every graph embeds on a 3-book (a complex made of three triangles sharing a common edge); thus, we only need to consider 2-complexes without 3-book, for otherwise the problem is trivial; this significantly restricts the structure of the 2-complexes to be considered. The problem of whether Embed admits an algorithm that is fixed-parameter tractable in terms of the parameter c, however, remains open for general complexes, whereas it is the case when restricting to surfaces [22].

Why is embedding graphs on 2-complexes interesting? First, let us remark that, if we consider the problem of embedding graphs into simplicial complexes, then the case that we consider, in which the complex has dimension at most two, is the only interesting one, since every graph can be embedded in a single tetrahedron.

We have already noted that the problem we study is more general than the problem of embedding graphs on surfaces. It is indeed quite general, and some other problems studied in the past can be recast as an instance of Embed or as variants of it. For example, the *crossing number* of a graph G is the minimum number of crossings in a (topological) drawing of G in the plane. Deciding whether a graph G has crossing number at most k is NP-hard, but fixed-parameter tractable in k, as shown by Kawarabayashi and Reed [14]. This is easily seen to be equivalent to the embeddability of G into the complex obtained by removing k disjoint disks from a sphere and adding, for each resulting boundary component b, two edges with endpoints on b whose cyclic order along b is interlaced. Of course, the embeddability problem on a 2-complex is more general and contains, for example, the problem of deciding whether there is a drawing of a graph G on a surface of genus g with at most k crossings. In topological graph theory, embeddings of graphs on pseudosurfaces (which are special 2-complexes) have been considered; see Archdeacon [2, Section 5.7] for a survey. Slightly more remotely, a *book embedding* of a graph G (see, e.g., Malik [18]) is also an embedding of G into a particular 2-complex, with additional constraints on the embedding.

Strategy of the proof and organization of the paper. For clarity of exposition, in most of the paper, we focus on developing an algorithm for the problem Embed (Theorem 2). Only at the end (Section 8) we explain why our techniques imply that the problem is in NP. The idea of the algorithm is to progressively reduce the problem to simpler problems. We first deal with the case where the complex $\mathscr{C}$ contains a 3-book (Section 3). From Section 4 onwards, we reduce Embed to *embedding extension problems* (EEP), similar to the Embed problem except that an embedding of a subgraph H of the input graph G is already specified. In Section 4, we reduce Embed to EEPs on a pure 2-complex (in which every segment of the complex $\mathscr{C}$ is incident to at least one triangle). In Section 5, we further reduce it to EEPs on a surface. In Section 6, we reduce it to EEPs on a surface in which every face of the subgraph H is a disk. Finally, in Section 7, we show how to solve EEPs of the latter type using a key component in an algorithm by the third author [22] to decide embeddability of a graph on a surface.

2 Preliminaries

2.1 Embeddings of graphs into 2-complexes

A **2-complex** is an abstract simplicial complex of dimension at most two: a finite set of 0-simplices called **nodes**, 1-simplices called **segments**, and 2-simplices called **triangles** (we use this terminology to distinguish from that of vertices and edges, which we reserve for graphs); each segment is a pair of nodes, and each triangle is a triple of nodes; moreover, each subset of size two in a triangle must be a segment. Each 2-complex $\mathscr{C}$ corresponds naturally to a topological space, obtained in the obvious way: Start with one point per node in $\mathscr{C}$; connect them by segments as indicated by the segments in $\mathscr{C}$; similarly, for every triangle in $\mathscr{C}$, create a triangle whose boundary is made of the three segments contained in that triangle. By abuse of language, we identify $\mathscr{C}$ with that topological space. To emphasize that we consider the abstract simplicial complex and not only the topological space, we sometimes use the name **triangulation** or **triangulated complex**.

In this paper, graphs are finite, undirected, and may have loops and multiple edges. In a similar way as for 2-complexes, each graph has an associated topological space; an **embedding** of a graph G into a 2-complex $\mathscr{C}$ is an injective continuous map from (the topological space associated to) G to (the topological space associated to) $\mathscr{C}$.

2.2 Structural aspects of 2-complexes

We say that a 2-complex **contains a 3-book** if some three distinct triangles share a common segment.

Let p be a node of $\mathscr{C}$. A **cone at** p is a cyclic sequence of triangles $t_1, \dots, t_k, t_{k+1} = t_1$, all incident to p, such that, for each $i = 1, \dots, k$, the triangles t_i and t_{i+1} share a segment incident with p, and any other pair of triangles have only p in common. A **corner at** p is an inclusionwise maximal sequence of triangles $t_1, \dots, t_k$, all incident to p, such that, for each $i = 1, \dots, k-1$, the triangles t_i and t_{i+1} share a segment incident with p, and any other pair of triangles have only p in common. An **isolated segment at** p is a segment incident to p but not incident to any triangle.

If $\mathscr{C}$ contains no 3-books, the set of segments and triangles incident with a given node p of $\mathscr{C}$ are uniquely partitioned into cones, corners, and isolated segments. We say that p is a **regular node** if all the segments and triangles incident to p form a single cone or corner. Otherwise, p is a **singular node**. A 2-complex is **pure** if it contains no isolated segment, and each node is incident to at least one segment.

2.3 Embedding extension problems and reductions

An **embedding extension problem** (EEP) is a decision problem defined as follows:

EEP(n, m, c):
Input: A graph G with n vertices and edges, a subgraph H of G with m vertices and edges, and an embedding Π of H into a 2-complex $\mathscr{C}$ with c simplices.
Question: Does G have an embedding into $\mathscr{C}$ whose restriction to H is Π?

To be precise, we will have to explain how we represent the embedding Π, but this will vary throughout the proof, and we will be more precise about this in subsequent sections. Let us simply remark that, since the complexity of our algorithm is a polynomial of large degree (depending on the complex $\mathscr{C}$) in the size of the input graph, the choice of representation is not very important, because converting between any two reasonable representations is possible in polynomial time.

We will reduce our original problem to more and more specialized EEPs. We will use the word "reduce" in a somewhat sloppy sense: A decision problem P **reduces** to k instances of the decision problem P' if solving these k instances of P' allows to solve the instance of P in time $O(k)$. We will have to be more precise when we consider the NP-completeness of Embed in Section 8.

2.4 Surfaces

In Section 6, we will assume some familiarity with surface topology; see, e.g., [5, 23, 27] for suitable introductions under various viewpoints. We recall some basic definitions and properties. A **surface** $\mathscr{S}$ is a compact, connected Hausdorff topological space in which every point has a neighborhood homeomorphic to the plane. Every surface $\mathscr{S}$ is obtained from a sphere by:

- either removing $g/2$ open disks and attaching a handle (a torus with an open disk removed) to each resulting boundary component, for an even, nonnegative number g called the *(Euler)* **genus** of $\mathscr{S}$; in this case, $\mathscr{S}$ is **orientable**;
- or removing g open disks and attaching a Möbius band to each resulting boundary component, for a positive number g called the **genus** of $\mathscr{S}$; in this case, $\mathscr{S}$ is **non-orientable**.

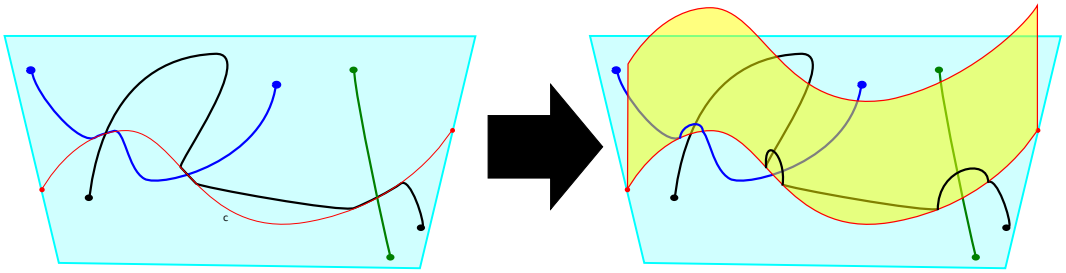


Figure 1 Illustration of the proof of Proposition 3. Left: The drawing of the graph G and the curve c (in thin). Right: The construction of the 3-book and the modification of the drawing.

A **surface with boundary** is obtained from a surface by removing a finite set of interiors of disjoint closed disks. The boundary of each disk forms a **boundary component** of $\mathscr{S}$. A **possibly disconnected surface** is a disjoint union of surfaces. An embedding of G into a surface $\mathscr{S}$, possibly with boundary, is **cellular** if each face of the embedding is homeomorphic to an open disk. If G is cellularly embedded on a surface with genus g and b boundary components, with v vertices, e edges, and f faces, then Euler's formula stipulates that $2 - g - b = v - e + f$.

An **ambient isotopy** of a surface with boundary $\mathscr{S}$ is a continuous family $(h_t)_{t\in[0,1]}$ of self-homeomorphisms of $\mathscr{S}$ such that h_0 is the identity.

3 Reduction to complexes containing no 3-book

The following folklore observation allows us to solve the problem trivially if $\mathscr{C}$ contains a 3-book. We include a proof for completeness.

▶ **Proposition 3.** *If $\mathscr{C}$ contains a 3-book, then every graph embeds into $\mathscr{C}$.*

Proof. Let G be a graph. We first draw G, possibly with crossings, in general position in the interior of a closed disk D. Let c be a simple curve in D with endpoints on ∂D and passing through all crossing points of the drawing of G. By perturbing c, we can ensure that, in the neighborhood of each crossing point of that drawing, c coincides with the image of one of the two edges involved in the crossing. See Figure 1, left.

Let D' be a closed disk disjoint from D. We attach D' to D by identifying c with a part of the boundary of D'. Now, in the neighborhood of each crossing of the drawing of G, we push inside D' the part of the edge coinciding with c, keeping its endpoints fixed. See Figure 1, right. This removes the crossings.

So G embeds in the topological space obtained from D by attaching a part of the boundary of D' along c. But this space embeds in $\mathscr{C}$, because $\mathscr{C}$ contains a 3-book. ◀

4 Reduction to EEPs on a pure 2-complex

Our next task is to reduce the problem Embed to a problem on a pure 2-dimensional complex. More precisely, let **EEP-Sing** be the problem EEP, restricted to instances $(G, H, \Pi, \mathscr{C})$ where: $\mathscr{C}$ is a pure 2-complex containing no 3-books; H is a set of vertices of G; and Π is an injective map from H to the nodes of $\mathscr{C}$ such that $\Pi(H)$ contains all singular nodes of $\mathscr{C}$. In this section, we prove the following result.

▶ **Proposition 4.** *Any instance of* Embed(n, c) *reduces to* $(cn)^{O(c)}$ *instances of* EEP-Sing$(cn, c, O(c))$.

First, a definition. Consider a map $f : P \to V(G) \cup \{\varepsilon\}$, where P is a set of nodes in $\mathscr{C}$ containing all singular nodes of $\mathscr{C}$. We say that an embedding Γ of G **respects** f if, for each $p \in P$, the following holds: If $f(p) = \varepsilon$, then p is not in the image of Γ; otherwise, $\Gamma(f(p)) = p$.

In this section, we will need the following intermediate problem:

Embed-Resp(n, m, c):
Input: A graph G with n vertices and edges, a 2-complex $\mathscr{C}$ (not necessarily pure) containing no 3-books, with c simplices, and a map f as above, with domain of size m.
Question: Does G have an embedding into $\mathscr{C}$ respecting f?

▶ **Lemma 5.** *Any instance of* Embed(n, c) *reduces to* $(O(cn))^c$ *instances of* Embed-Resp(cn, c, c).

Proof. By Proposition 3, we can without loss of generality assume that $\mathscr{C}$ contains no 3-books. Let G' be the graph obtained from G by subdividing each edge k times, where $k \leq c$ is the number of singular nodes of $\mathscr{C}$. We claim that G has an embedding into $\mathscr{C}$ if and only if G' has an embedding Γ' into $\mathscr{C}$ such that each singular node of $\mathscr{C}$ in the image of Γ' is the image of a vertex of G'.

Indeed, assume that G has an embedding Γ on $\mathscr{C}$. Each time an edge of G is mapped, under Γ, to a singular node p of $\mathscr{C}$, we subdivide this edge and map this new vertex to p; the image of the embedding is unchanged. This ensures that only vertices are mapped to singular nodes. Moreover, there were at most k subdivisions, one per singular node. So, by further subdividing the edges until each original edge is subdivided k times, we obtain an embedding of G' to $\mathscr{C}$ such that only vertices are mapped on singular nodes. The reverse implication is obvious: If G' has an embedding into $\mathscr{C}$, then so has G. This proves the claim.

To conclude, for each map from the set of singular vertices of $\mathscr{C}$ to $V(G') \cup \{\varepsilon\}$, we solve the problem whether G' has an embedding on $\mathscr{C}$ respecting f. The graph G embeds on $\mathscr{C}$ if and only if the outcome is positive for at least one such map f. By construction, there are at most $(kn + 1)^k = (O(nc))^c$ such maps, because $V(G')$ has size at most kn. ◀

▶ **Lemma 6.** Embed-Resp(n, m, c) *reduces to* EEP-Sing$(n, m, O(c))$.

Proof. We omit the proof due to space constraints, but the idea is simple:

Because we restrict ourselves to embeddings that respect f, and thus specify which vertex of G is mapped to each of the singular nodes of $\mathscr{C}$, what happens on the isolated segments of $\mathscr{C}$ is essentially determined. ◀

Before giving the proof, we first note:

Proof of Proposition 4. It follows immediately from Lemmas 5 and 6. ◀

5 Reduction to an EEP on a possibly disconnected surface

The previous section led us to an embedding extension problem in a pure 2-complex without 3-book where the images of some vertices are predetermined. Now, we show that solving such an EEP amounts to solving another EEP in which the complex is a surface.

Let **EEP-Surf** be the problem EEP, restricted to instances where the input complex is (homeomorphic to) a possibly disconnected triangulated surface without boundary (which we denote by $\mathscr{S}$ instead of $\mathscr{C}$, for clarity). To represent the embedding Π in such an EEP instance $(G, H, \Pi, \mathscr{S})$, it will be convenient to use the fact that, in all our constructions below, the image of every connected component of H under Π will intersect the 1-skeleton

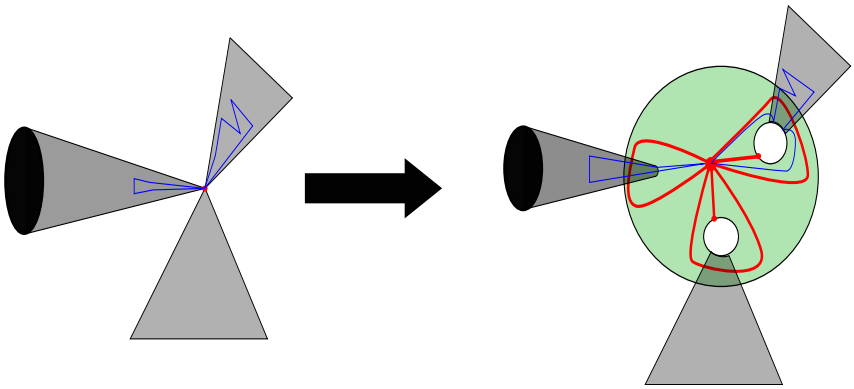


Figure 2 The modification of singular vertices in the proof of Lemma 8. We transform the neighborhood of each singular vertex to make it surface-like. Moreover, we add to H one loop per cone or corner; furthermore, for each corner, we add a vertex and an edge.

of $\mathscr{S}$ at least once, and finitely many times. (Note that H may use some nodes of $\mathscr{S}$.) Consider the *overlay* of the triangulation of $\mathscr{S}$ and of Π, the union of the 1-skeleton of $\mathscr{S}$ and of the image of Π; this overlay is the image of a graph on $\mathscr{S}$; each of its edges is either a piece of the image of an edge of H or a piece of a segment of $\mathscr{S}$; each of its vertices is the image of a vertex of H and/or a node of $\mathscr{S}$. By the assumption above on Π, this overlay is cellularly embedded on $\mathscr{S}$, and we can represent it by its combinatorial map [10,17] (possibly on surfaces with boundary, since at intermediary steps of our construction we will have to consider such surfaces).

In this section, we prove the following proposition.

▶ **Proposition 7.** *Any instance of* EEP-Sing(n, m, c) *reduces to an instance of* EEP-Surf$(O(n+m+c), O(m+c), O(c))$.

We will first reduce the original EEP to an intermediary EEP on a surface with boundary.

▶ **Lemma 8.** *Any instance of* EEP-Sing(n, m, c) *reduces to an instance of* EEP$(n + O(c), m + O(c), O(c))$ *in which the considered 2-complex is a possibly disconnected surface with boundary.*

Proof. The key property that we will use is that, since $\mathscr{C}$ is pure and contains no 3-books, each singular node is incident to cones and corners only.

Figure 2 illustrates the proof. Let $(G, H, \Pi, \mathscr{C})$ be the instance of EEP-Sing. We first describe the construction of the instance $(G', H', \Pi', \mathscr{S})$ on the possibly disconnected surface with boundary. Let p be a singular node; we modify the complex in the neighborhood of p as follows. Let c_p be the number of cones at p and c'_p be the number of corners at p. We remove a small open neighborhood N_p of p from $\mathscr{C}$, in such a way that the boundary of N_p is a disjoint union of c_p circles and c'_p arcs. We create a sphere S_p with $c_p + c'_p$ boundary components. Finally, we attach each circle and arc to a different boundary component of S_p, choosing an arbitrary orientation for each gluing; circles are attached to an entire boundary component of S_p, while arcs cover only a part of a boundary component of S_p. Doing this for every singular node p, we obtain a surface (possibly disconnected, possibly with boundary), which we denote by $\mathscr{S}$.

We now define H', G', and Π' from H, G, and Π (again, refer to Figure 2). Let p be a singular node of $\mathscr{C}$ and v_p the vertex of H mapped on p by Π. In H (and thus also G), we

add a set L_p of $c_p + c'_p$ loops with vertex v_p. Let q_p be a point in the interior of S_p; in Π, we map v_p to q_p, and we map these $c_p + c'_p$ loops on S_p in such a way that each loop encloses a different boundary component of S_p (thus, if we cut S_p along these loops, we obtain $c_p + c'_p$ annuli and one disk).

Finally, we add to H (and thus also to G) a set E_p of c'_p new edges, each connecting v_p to a new vertex. In Π, each new vertex is mapped to the boundary component of S_p corresponding to a corner, but not on the corresponding arc.

Let us call G' and H' the resulting graphs, and Π' the resulting embedding of H'. Note that, from the triangulation of $\mathscr{C}$ with c simplices, we can easily obtain a triangulation of $\mathscr{S}$ with $O(c)$ simplices, and that the image of each edge of H crosses $O(1)$ edges of this triangulation.

There remains to prove that these two EEPs are equivalent; we omit the details. ◀

We now deduce from the previous EEP the desired EEP on a surface without boundary.

▶ **Lemma 9.** *Any instance of* EEP-Sing(n, m, c) *on a possibly disconnected surface with boundary reduces to an instance of* EEP-Surf$(n + O(m), O(m), O(c))$.

Proof. Let $(G, H, \Pi, \mathscr{S})$ be an instance of an EEP on a possibly disconnected surface with boundary. We first describe the construction of $(G', H', \Pi', \mathscr{S}')$, the EEP instance on a possibly disconnected surface without boundary.

Let $\mathscr{S}'$ be obtained from $\mathscr{S}$ by gluing a disk D_b along each boundary component. Let b be a boundary component of $\mathscr{S}$. If Π maps at least one vertex to b, then we add to H (and thus also to G) a new vertex v_b, which we connect, also by a new edge, to each of the vertices mapped to b by Π. We extend Π by mapping vertex v_b and its incident edges inside D_b. Let us call G' and H' the resulting graphs, and Π' the resulting embedding of H'. For each vertex of H on a boundary component, we added to H and G at most one vertex and one edge. There remains to prove that these two EEPs are equivalent; we omit the details. ◀

Finally:

Proof of Proposition 7. It suffices to successively apply Lemmas 8 and 9. ◀

6 Reduction to a cellular EEP on a surface

Let **EEP-Cell** be the problem EEP, restricted to instances $(G, H, \Pi, \mathscr{S})$ where $\mathscr{S}$ is a surface and H is cellularly embedded and intersects each connected component of G.

In this section, we prove the following proposition.

▶ **Proposition 10.** *Any instance of* EEP-Surf(n, m, c) *reduces to* $(n+m+c)^{O(m+c)}$ *instances of* EEP-Cell$(O(n + m + c), O(n + m + c), c)$.

As will be convenient also for the next section, we do not store an embedding Π of a graph G on a surface $\mathscr{S}$ by its overlay with the triangulation, as was done in the previous section, but we forget the triangulation. In other words, we have to store the combinatorial map corresponding to Π, but taking into account the fact that Π is not necessary cellular: We need to store, for each face of the embedding, whether it is orientable or not, and a pointer to an edge of each of its boundary components (with some orientation information). Such a data structure is known under the name of *extended combinatorial map* [6, Section 2.2] (only orientable surfaces were considered there, but the data structure readily extends to non-orientable surfaces).

6.1 Reduction to connected surfaces

We first build intermediary EEPs over connected surfaces. Let **EEP-Conn** be the problem EEP, restricted to instances $(G, H, \Pi, \mathscr{S})$ where $\mathscr{S}$ is a surface (connected and without boundary) and H intersects every connected component of G.

▶ **Lemma 11.** *Any instance of* EEP-Surf(n, m, c) *reduces to* $O(m^c)$ *instances of* EEP-Conn$(n, m + c, c)$.

More precisely (and this is a fact that will be useful to prove that Embed is in NP, see Theorem 1), any instance of EEP-Surf(n, m, c) is equivalent to the disjunction (OR) of $O(m^c)$ instances, each of them being the conjunction (AND) of $O(c)$ instances of EEP-Conn$(n, m + O(c), c)$.

Sketch of proof. We can embed each connected component of G that is planar and disjoint from H anywhere. There remains $O(c)$ connected components of G that are disjoint from H. For each of these, we choose a vertex, and we guess in which face of Π it belongs. We then know which connected component of the surface each connected component of G is mapped into. ◀

6.2 The induction

The strategy for the proof of Proposition 10 is as follows. For each EEP $(G', H', \Pi', \mathscr{S}')$ from the previous lemma, we will extend H' to make it cellular, by adding either paths connecting two boundary components of a face of H', or paths with endpoints on the same boundary component of a face of H' in a way that the genus of the face decreases. We first define an invariant that will allow to prove that this process terminates.

Let Π be an embedding of a graph H on a surface $\mathscr{S}$. The **cellularity defect** of $(H, \Pi, \mathscr{S})$ is the non-negative integer

$$\mathbf{cd}(H, \Pi, \mathscr{S}) := \sum_{f \text{ face of } \Pi} \text{genus}(f) + \sum_{f \text{ face of } \Pi} (\text{number of boundaries of } f - 1).$$

Some obvious remarks: Π can contain isolated vertices; by convention, each of them counts as a boundary component of the face of Π it lies in. With this convention, every face of H has at least one boundary component, except in the very trivial case where G is empty. This implies that Π is a cellular embedding if and only if $cd(H, \Pi, \mathscr{S}) = 0$.

The following lemma reduces an EEP to EEPs with a smaller cellularity defect.

▶ **Lemma 12.** *Any instance of* EEP-Conn$(n, O(n), c)$ *reduces to* $O(n^4)$ *instances* $(G', H', \Pi', \mathscr{S})$ *of* EEP-Conn$(n + O(1), O(n), c)$ *where* $cd(H', \Pi', \mathscr{S}) < cd(H, \Pi, \mathscr{S})$.

The reduction does not depend on the size of H; furthermore, the new graph G' is obtained from the old one by adding exactly one edge and no vertex.

Admitting Lemma 12, the proof of Proposition 10 is straightforward:

Proof of Proposition 10. We first apply Lemma 11, obtaining $O(m^c)$ instances of EEP-Conn$(n, m + c, c)$. To each of these EEPs, we apply recursively Lemma 12 until we obtain cellular EEPs. The cellularity defect of the initial instance $(G, H, \Pi, \mathscr{S})$ is $O(m + c)$, being at most the genus of $\mathscr{S}$ plus $2m$, because each boundary component of a face of Π is incident to at least one edge of H (and each edge accounts for at most two boundary components in this way) or to one isolated vertex of H. Thus, the number of instances of EEP-Cell at the bottom of the recursion tree is $(n + m + c)^{O(m+c)}$, in which the size of the graph is $O(n + m + c)$ and the surface has at most c simplices. ◀

6.3 Proof of Lemma 12

There remains to prove Lemma 12. The proof uses some standard notions in surface topology, homotopy, and homology; we refer to textbooks and surveys [5, 23, 27]. We only consider homology with $\mathbb{Z}/2\mathbb{Z}$ coefficients.

Let f be a surface with a single boundary component and let p be a path with endpoints on the boundary of f. If we contract this boundary component to a single point, the path p becomes a loop, which can be null-homologous or non-null-homologous. We employ the same adjectives null-homologous and non-null-homologous for the path p. Recall that, if p is simple, it separates f if and only if it is null-homologous. The reversal of a path p is denoted by $\bar{p}$. The concatenation of two paths p and q is denoted by $p \cdot q$.

▶ **Lemma 13.** *Let f be a surface with boundary, let a be a point in the interior of f, and let a_1, a_2, and a_3 be points on the boundary of f. For each i, let p_i be a path connecting a_i to a. Let $r_1 = p_2 \cdot \bar{p}_3$, $r_2 = p_3 \cdot \bar{p}_1$, and $r_3 = p_1 \cdot \bar{p}_2$. Let P be a possible property of the paths r_i, among the following ones:*

- *"the endpoints of r_i lie on the same boundary component of f";*
- *"r_i is null-homologous" (if f has a single boundary component).*

Then the following holds: If both r_1 and r_2 have property P, then so does r_3.

Proof. This is a variant on the *3-path condition* from Mohar and Thomassen [23, Section 4.3]. The first item is immediate. The second one follows from the fact that homology is an algebraic condition: The concatenation of two null-homologous paths is null-homologous, and removing subpaths of the form $q \cdot \bar{q}$ from a path does not affect homology. ◀

Proof of Lemma 12. Since $cd(H, \Pi, \mathscr{S}) \geq 1$, there must be a face f of H with either (1) several boundary components, or (2) a single boundary component but positive genus. We will consider each of these cases separately, but first introduce some common terminology.

Let F be an arbitrary spanning forest of $G - E(H)$ rooted at $V(H)$. This means that F is a subgraph of $G - E(H)$ that is a forest with vertex set $V(G)$ such that each connected component of F contains exactly one vertex of $V(H)$, its *root*. The algorithm starts by computing an arbitrary such forest F in linear time.

For each vertex u of G, let $r(u)$ be the unique root in the same connected component of F as u, and let $F(u)$ be the unique path connecting u to $r(u)$. If u and v are two vertices of G, let G_{uv} be the graph obtained from G by adding one edge, denoted uv, connecting u and v. (This may be a parallel edge if u and v were already adjacent in G, but in such a situation when we talk about edge uv we always mean the new edge.) Let $F(uv)$ be the unique path between u and v in G that is the concatenation of $\overline{F(u)}$, edge uv, and $F(v)$.

Case 1: f has several boundary components. Assume that $(G, H, \Pi, \mathscr{S})$ has a solution Γ. We claim that, *for some vertices u and v of G, the embedding Γ extends to an embedding of G_{uv} in which the image of the path $F(uv)$ lies in f and connects two distinct boundary components of f.*

Indeed, let c be a curve drawn in f connecting two different boundary components of f. We can assume that it intersects the boundary of f exactly at its endpoints, at vertices of H. We can deform c so that it intersects Γ only at the images of vertices, and never in the relative interior of an edge. We can, moreover, assume that c is simple (except perhaps that its endpoints coincide on $\mathscr{S}$). Let $v_1, \ldots, v_k$ be the vertices of G encountered by c, in this order. We denote by $c[i, j]$ the part of c between vertices v_i and v_j. We claim that, for some i, we have that $\overline{F(v_i)} \cdot c[i, i+1] \cdot F(v_{i+1})$ connects two different boundary components

of f: Otherwise, by induction on i, applying the first case of Lemma 13 to the three paths $\overline{c[1,i]}$, $F(v_i)$, and $c[i,i+1] \cdot F(v_{i+1})$, we would have that, for each i, $c[1,i] \cdot F(v_i)$ has its endpoints on the same boundary component of f, which is a contradiction for $i = k$ (for which the curve is c). So let i be such that $\overline{F(v_i)} \cdot c[i,i+1] \cdot F(v_{i+1})$ connects two different boundary components of f; letting $u = v_i$ and $v = v_{i+1}$, and embedding edge uv as $c[i,i+1]$, gives the desired embedding of G_{uv}. This proves the claim.

The strategy now is to guess the vertices u and v and the way the path $F(uv)$ is drawn in f, and to solve a set of EEPs $(G_{uv}, H \cup F(uv), \Pi', \mathscr{S})$ where Π' is chosen as an appropriate extension of Π. *Let us first assume that f is orientable.* One subtlety is that, given u and v, there can be several essentially different ways of embedding $F(uv)$ inside f, if there is more than one occurrence of $r(u)$ and $r(v)$ on the boundary of f. So we reduce our EEP to the following set of EEPs: For each choice of vertices u and v of G, and each occurrence of $r(u)$ and $r(v)$ on the boundary of f, we consider the EEP $(G_{uv}, H \cup F(uv), \Pi', \mathscr{S})$ where Π' extends Π and maps $F(uv)$ to an arbitrary path in f connecting the chosen occurrences of $r(u)$ and $r(v)$ on the boundary of f.

It is clear that, if one of these new EEPs has a solution, the original EEP has a solution. Conversely, let us assume that the original EEP $(G, H, \Pi, \mathscr{S})$ has a solution; we now prove that one of these new EEPs has a solution. By our claim above, for some choice of u and v, some EEP $(G_{uv}, H \cup F(uv), \Pi'', \mathscr{S})$ has a solution, for some Π'' mapping $F(uv)$ inside f and connecting different boundary components of f. In that mapping, $F(uv)$ connects two occurrences of $r(u)$ and $r(v)$ inside f. We prove that, for these choices of occurrences of $r(u)$ and $r(v)$, the corresponding EEP described in the previous paragraph, $(G_{uv}, H \cup F(uv), \Pi', \mathscr{S})$, has a solution as well. These two EEPs are the same except that the path $F(uv)$ may be drawn differently in Π' and Π'', although they connect the same occurrences of $r(u)$ and $r(v)$ on the boundary of f. Under Π', the face f is transformed into a face f' that has the same genus and orientability character as f, but one boundary component less. The same holds, of course, for Π''. Moreover, the ordering of the vertices on the boundary components of the new face is the same in Π' and Π''. Thus, there is a homeomorphism h of f that keeps the boundary of f fixed pointwise and such that $h \circ \Pi''|_{F(uv)} = \Pi'|_{F(uv)}$. This homeomorphism, extended to the identity outside f, maps any solution of $(G_{uv}, H \cup F(uv), \Pi'', \mathscr{S})$ to a solution of $(G_{uv}, H \cup F(uv), \Pi', \mathscr{S})$, as desired.

It also follows from the previous paragraph that the cellularity defect decreases by one. To conclude this case, we note that the number of new EEPs is $O(n^4)$: indeed, there are $O(n)$ possibilities for the choice of u (or v), and $O(n)$ possibilities for the choice of the occurrence of $r(u)$ (or $r(v)$) on the boundary of f.

If f is non-orientable, the same argument works, except that there are two possibilities for the cyclic ordering of the vertices along the new boundary component of the new face: If we walk along one of the boundary components of f (in an arbitrary direction), use p, and walk along the other boundary component of f, we do not know in which direction this second boundary component is visited. So we actually need to consider two EEPs for each choice of u, v, and occurrences of $r(u)$ and $r(v)$, instead of one. The rest is unchanged.

Case 2: f has a single boundary component and positive genus. The proof is very similar to the previous case, the main difference being that, instead of paths in f connecting different boundary components of f, we now consider paths in f that are non-null-homologous. ◀

7 Solving a cellular EEP on a surface

▶ **Proposition 14.** *There is a function $f : \mathbb{N} \to \mathbb{N}$ such that every instance of* EEP-CELL$(n, O(n), c)$ *can be solved in time $f(c) \cdot O(n)$.*

Proof. This is essentially the main result of Mohar [22]. The algorithm in [22] makes reductions to even more specific EEPs, and one feature that is needed for bounding the number of new instances is that the embedded subgraph H satisfies some connectivity assumptions. ◀

8 Proof of Theorems 1 and 2

We can finally prove our main theorems. First, let us prove that we have an algorithm with complexity $f(c) \cdot n^{O(c)}$:

Proof of Theorem 2. This immediately follows from Propositions 3, 4, 7, 10, and 14. ◀

Finally, we prove that the EMBED problem is NP-complete:

Proof of Theorem 1. The problem EMBED is NP-hard because deciding whether an input graph embeds into an input surface is NP-hard [28]. It is in NP because (assuming $\mathscr{C}$ contains no 3-books), a certificate that an instance is positive is given by a certificate that some instance of EEP-SURF is positive (see the proof of Proposition 7). Such an instance $(G, H, \Pi, \mathscr{S})$ has a certificate, given by the combinatorial map of a supergraph G' of G cellularly embedded on $\mathscr{S}$ (see Section 6), that can be checked in polynomial time. ◀

References

1. Patrizio Angelini, Giuseppe Di Battista, Fabrizio Frati, Vít Jelínek, Jan Kratochvíl, Maurizio Patrignani, and Ignaz Rutter. Testing planarity of partially embedded graphs. *ACM Transactions on Algorithms (TALG)*, 11(4):32, 2015.
2. Dan Archdeacon. Topological graph theory. A survey. *Congressus Numerantium*, 115:5–54, 1996.
3. William W. Boone. The word problem. *Annals of Mathematics*, 70:207–265, 1959.
4. Martin Čadek, Marek Krčál, Jiří Matoušek, Lukáš Vokřínek, and Uli Wagner. Polynomial-time computation of homotopy groups and Postnikov systems in fixed dimension. *SIAM Journal on Computing*, 43(5):1728–1780, 2014.
5. Éric Colin de Verdière. Computational topology of graphs on surfaces. In Jacob E. Goodman, Joseph O'Rourke, and Csaba Toth, editors, *Handbook of Discrete and Computational Geometry*, chapter 23. CRC Press LLC, third edition, 2018. See `http://www.arxiv.org/abs/1702.05358`.
6. Éric Colin de Verdière and Arnaud de Mesmay. Testing graph isotopy on surfaces. *Discrete & Computational Geometry*, 51(1):171–206, 2014.
7. Erik Demaine, Shay Mozes, Christian Sommer, and Siamak Tazari. Algorithms for planar graphs and beyond, 2011. Course notes available at `http://courses.csail.mit.edu/6.889/fall11/lectures/`.
8. Tamal K. Dey, Herbert Edelsbrunner, and Sumanta Guha. Computational topology. In Bernard Chazelle, Jacob E. Goodman, and Richard Pollack, editors, *Advances in Discrete and Computational Geometry – Proc. 1996 AMS-IMS-SIAM Joint Summer Research Conf. Discrete and Computational Geometry: Ten Years Later*, number 223 in Contemporary Mathematics, pages 109–143. AMS, 1999.

9 Tamal K. Dey and Sumanta Guha. Transforming curves on surfaces. *Journal of Computer and System Sciences*, 58:297–325, 1999.

10 David Eppstein. Dynamic generators of topologically embedded graphs. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 599–608, 2003.

11 Jeff Erickson. Combinatorial optimization of cycles and bases. In Afra Zomorodian, editor, *Computational topology*, Proceedings of Symposia in Applied Mathematics. AMS, 2012.

12 Jeff Erickson and Kim Whittlesey. Transforming curves on surfaces redux. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1646–1655, 2013.

13 Ken-ichi Kawarabayashi, Bojan Mohar, and Bruce Reed. A simpler linear time algorithm for embedding graphs into an arbitrary surface and the genus of graphs of bounded tree-width. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 771–780, 2008.

14 Ken-ichi Kawarabayashi and Bruce Reed. Computing crossing number in linear time. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC)*, pages 382–390, 2007.

15 Tomasz Kociumaka and Marcin Pilipczuk. Deleting vertices to graphs of bounded genus. arXiv:1706.04065, 2017.

16 Francis Lazarus and Julien Rivaud. On the homotopy test on surfaces. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 440–449, 2012.

17 Sóstenes Lins. Graph-encoded maps. *Journal of Combinatorial Theory, Series B*, 32:171–181, 1982.

18 Seth M. Malitz. Genus g graphs have pagenumber $O(\sqrt{g})$. *Journal of Algorithms*, 17:85–109, 1994.

19 Jiří Matoušek, Eric Sedgwick, Martin Tancer, and Uli Wagner. Embeddability in the 3-sphere is decidable. *Journal of the ACM*, 2018. To appear. Preliminary version in *Symposium on Computational Geometry 2014*.

20 Jiří Matoušek, Martin Tancer, and Uli Wagner. Hardness of embedding simplicial complexes in R^d. *Journal of the European Mathematical Society*, 13(2):259–295, 2011.

21 Bojan Mohar. On the minimal genus of 2-complexes. *Journal of Graph Theory*, 24(3):281–290, 1997.

22 Bojan Mohar. A linear time algorithm for embedding graphs in an arbitrary surface. *SIAM Journal on Discrete Mathematics*, 12(1):6–26, 1999.

23 Bojan Mohar and Carsten Thomassen. *Graphs on surfaces*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 2001.

24 Colm Ó Dúnlaing, Colum Watt, and David Wilkins. Homeomorphism of 2-complexes is equivalent to graph isomorphism. *International Journal of Computational Geometry & Applications*, 10:453–476, 2000.

25 Maurizio Patrignani. Planarity testing and embedding. In Roberto Tamassia, editor, *Handbook of graph drawing and visualization*. Chapman and Hall, 2006.

26 Neil Robertson and Paul D. Seymour. Graph minors. XIII. The disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 63(1):65–110, 1995.

27 John Stillwell. *Classical topology and combinatorial group theory*. Springer-Verlag, New York, second edition, 1993.

28 Carsten Thomassen. The graph genus problem is NP-complete. *Journal of Algorithms*, 10(4):568–576, 1989.

On the Complexity of Closest Pair via Polar-Pair of Point-Sets

Roee David[1]
Datorama
Israel
daroeevid@gmail.com

Karthik C. S.[2]
Weizmann Institute of Science
Rehovot, Israel
karthik.srikanta@weizmann.ac.il

Bundit Laekhanukit[3]
Shanghai University of Finance and Economics
Shanghai, China
Max-Planck-Institute for Informatics
Saabrücken, Germany
blaekhan@mpi-inf.mpg.de

Abstract

Every graph G can be represented by a collection of equi-radii spheres in a d-dimensional metric Δ such that there is an edge uv in G if and only if the spheres corresponding to u and v intersect. The smallest integer d such that G can be represented by a collection of spheres (all of the same radius) in Δ is called the *sphericity* of G, and if the collection of spheres are non-overlapping, then the value d is called the *contact-dimension* of G. In this paper, we study the sphericity and contact dimension of the complete bipartite graph $K_{n,n}$ in various L^p-metrics and consequently connect the complexity of the monochromatic closest pair and bichromatic closest pair problems.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Contact dimension, Sphericity, Closest Pair, Fine-Grained Complexity

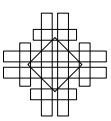
Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.28

Related Version A full version of this paper is available at `http://arxiv.org/abs/1608.03245`

Acknowledgements We would like to thank Aviad Rubinstein for sharing with us [26] and for the many discussions. We would like to thank Petteri Kaski and Rasmus Pagh for useful discussions and also for pointing out the reference [3]. We would like to thank Eylon Yogev and Amey Bhangale for some preliminary discussions. We would like to thank Uriel Feige for a lot of useful comments and discussions. Finally, we would like to thank Roei Tell for helping us improving the presentation of the paper.

[1] This work was partially supported by the ISF grant # 621/12) and by the I-CORE program grant # 4/11).
[2] This work was partially supported by Irit Dinur's ERC-StG grant # 239985 and ERC-CoG grant # 772839.
[3] This work was partially supported by the ISF grant # 621/12 and by the I-CORE program grant # 4/11). Part of this work was done while the third author was visiting the Simons Institute for the Theory of Computing. It was partially supported by the DIMACS/Simons Collaboration on Bridging Continuous and Discrete Optimization through the NSF grant # CCF-1740425.

34th International Symposium on Computational Geometry (SoCG 2018).
Editors: Bettina Speckmann and Csaba D. Tóth; Article No. 28; pp. 28:1–28:15



Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

This paper studies the geometric representation of a complete bipartite graph in L^p-metrics and consequently connects the complexity of the closest pair and bichromatic closest pair problems beyond certain dimensions. Given a point-set P in a d-dimensional L^p-metric, an *α-distance graph* is a graph $G = (V, E)$ with a vertex set $V = P$ and an edge set

$$E = \{uv : \|u - v\|_p \leq \alpha; u, v \in P; u \neq v\}.$$

In other words, points in P are centers of spheres of radius $\alpha/2$, and G has an edge uv if and only if the spheres centered at u and v intersect. The *sphericity* of a graph G in an L^p-metric, denoted by $\mathsf{sph}_p(G)$, is the smallest dimension d such that G is isomorphic to some α-distance graph in a d-dimensional L^p-metric, for some constant $\alpha > 0$. The sphericity of a graph in the L^∞-metric is known as *cubicity*. A notion closely related to sphericity is *contact-dimension*, which is defined in the same manner except that the spheres representing G must be non-overlapping. To be precise, an *α-contact graph* $G = (V, E)$ of a point-set P is an α-distance graph of P such that every edge uv of G has the same distance (i.e., $\|u - v\|_p = \alpha$). Thus, G has the vertex set $V = P$ and has an edge set E such that

$$\forall uv \in E, \quad \|u - v\|_p = \alpha \quad \text{and} \quad \forall uv \notin E, \quad \|u - v\|_p > \alpha.$$

The contact-dimension of a graph G in the L^p-metric, denoted by $\mathsf{cd}_p(G)$, is the smallest integer $d \geq 1$ such that G is isomorphic to a contact-graph in the d-dimensional L^p-metric. We will use distance and contact graphs to means 1-distance and 1-contact graphs.

We are interested in determining the sphericity and the contact-dimension of the biclique $K_{n,n}$ in various L^p-metrics. For notational convenience, we denote $\mathsf{sph}_p(K_{n,n})$ by $\mathsf{bsph}(L^p)$, the *biclique sphericity* of the L^p-metric, and denote $\mathsf{cd}_p(K_{n,n})$ by $\mathsf{bcd}(L^p)$, the *biclique contact-dimension* of the L^p-metric. We call a pair of point-sets (A, B) *polar* if it is the partition of the vertex set of a contact graph isomorphic to $K_{n,n}$. More precisely, a pair of point-sets (A, B) is polar in an L^p-metric if there exists a constant $\alpha > 0$ such that every inner-pair $u, u' \in A$ (resp., $v, v' \in B$) has L^p-distance greater than α while every crossing-pair $u \in A, v \in B$ has L^p-distance exactly α.

The biclique sphericity and contact-dimension of the L^2 and L^∞ metrics are well-studied in literature (see [25, 21, 22, 11, 23, 9]). Maehara [23, 21] showed that $n < \mathsf{bsph}(L^2) \leq (1.5)n$, and Maehara and Frankl & Maehara [22, 11] showed that $(1.286)n - 1 < \mathsf{bcd}(L^2) < (1.5)n$. For cubicity, Roberts [25] showed that $\mathsf{bcd}(L^\infty) = \mathsf{bsph}(L^\infty) = 2\log_2 n$. Nevertheless, for other L^p-metrics, contact dimension and sphericity are not well-studied.

1.1 Our results and contributions

Our main conceptual contribution is connecting the complexity of the (monochromatic) closest pair problem (Closest Pair) to that of the bichromatic closest pair problem (BCP) through the contact dimension of the biclique. This is discussed in subsection 1.1.1. Our main technical contributions are bounds on the contact dimension and sphericity of the biclique for various L^p-metrics. This is discussed in subsection 1.1.2. Finally, as an application of the connection discussed in subsection 1.1.1 and the bounds discussed in subsection 1.1.2, we show computational equivalence between monochromatic and bichromatic closest pair problems.

1.1.1 Connection between Closest Pair and BCP

In Closest Pair, we are asked to find a pair of points in a set of m points with minimum distance. BCP is a generalization of Closest Pair, in which each point is colored red or blue, and we are asked to find a pair of red-blue points (i.e., bichromatic pair) with minimum distance. It is not hard to see that BCP is at least as hard as Closest Pair since we can apply an algorithm for BCP to solve Closest Pair with the same asymptotic running time. However, it is not clear whether the other direction is true. We will give a simple reduction from BCP to Closest Pair using a polar-pair of point-sets. First, take a polar-pair (A, B), each with cardinality $n = m/2$, in the L^p-metric. Next, pair up vectors in A and B to red and blue points, respectively, and then attach a vertex $u \in A$ (resp., $v \in B$) to its matching red (resp., blue) point. This reduction increases the distances between every pair of points, but by the definition of the polar-pair, this process has more effect on the distances of the *monochromatic* (i.e., red-red or blue-blue) pairs than that of *bichromatic pairs*, and the reduction, in fact, has no effect on the order of crossing-pair distances at all. By scaling the vectors in A and B appropriately, this gives an instance of Closest Pair whose closest pair of points is bichromatic. Consequently, provided that the polar-pair of point-sets (A, B) in a d-dimensional metric can be constructed within a running time at least as fast as the time for computing Closest Pair in the same metric, this gives a reduction from BCP to Closest Pair, thus implying that they have the same running time lower bound.

1.1.2 Bounds on contact dimension and sphericity of biclique

Our main technical results are lower and upper bounds on the biclique contact-dimension for the L^p-metric space where $p \in \mathbb{R}_{\geq 1} \cup \{0\}$.

▶ **Theorem 1.** *The following are upper and lower bounds on biclique contact-dimension for the L^p-metric.*

$$\mathsf{bsph}(L^0) = \mathsf{bcd}(L^0) = n \tag{1}$$

$$n \leq \mathsf{bsph}(L^0_{\{0,1\}})) \leq \mathsf{bcd}(L^0_{\{0,1\}}) \leq n^2 \qquad \textit{(i.e., } P \subseteq \{0,1\}^d\textit{)} \tag{2}$$

$$\Omega(\log n) \leq \mathsf{bsph}(L^1) \leq \mathsf{bcd}(L^1) \leq n^2 \tag{3}$$

$$\Omega(\log n) \leq \mathsf{bsph}(L^p) \leq \mathsf{bcd}(L^p) \leq 2n \qquad \textit{for } p \in (1,2) \tag{4}$$

$$\mathsf{bsph}(L^p) = \Theta(\mathsf{bcd}(L^p)) = \Theta(\log n) \qquad \textit{for } p > 2 \tag{5}$$

Note that $\mathsf{bsph}(\Delta) \leq \mathsf{bcd}(\Delta)$ for any metric Δ. Thus, it suffices to prove a lower bound for $\mathsf{bsph}(\Delta)$ and prove an upper bound for $\mathsf{bcd}(\Delta)$.

We note that the bounds on the sphericity and the contact dimension of the L^1-metric in (3) are obtained from (5) and (1), respectively. We are unable to show a strong (e.g., linear) lower bound for the L^1-metric. However, we prove the weaker (average-case) result below for the L^1-metric which can be seen as a progress toward proving stronger lower bounds on the sphericity of the biclique in this metric (see Corollary 7 for more discussion on its applications).

▶ **Theorem 2.** *For any integer $d > 0$, there exist no two finite-supported random variables X, Y taking values from $\mathbb{R}^d$ such that the following hold.*

$$\mathop{\mathbb{E}}_{x_1,x_2 \in_R X}[\|x_1 - x_2\|_1] > \mathop{\mathbb{E}}_{x_1 \in_R X,\, y_1 \in_R Y}[\|x_1 - y_1\|_1]$$

$$\mathop{\mathbb{E}}_{y_1,y_2 \in_R Y}[\|y_1 - y_2\|_1] > \mathop{\mathbb{E}}_{x_1 \in_R X,\, y_1 \in_R Y}[\|x_1 - y_1\|_1] \ .$$

Table 1 Known bounds on sphericity and contact dimension of biclique.

Metric	Bound	From
L^0	$\mathsf{bsph}(L^0) = \mathsf{bcd}(L^0) = n$	This paper
L^1	$\Omega(\log n) \leq \mathsf{bsph}(L^1) \leq \mathsf{bcd}(L^1) \leq n^2$	This paper
L^p, $p \in (1,2)$	$\Omega(\log n) \leq \mathsf{bsph}(L^p) \leq \mathsf{bcd}(L^p) \leq 2n$	This paper
L^2	$n < \mathsf{bsph}(L^2) \leq \mathsf{bcd}(L^2) < 1.5 \cdot n$	[23, 11]
L^p, $p > 2$	$\mathsf{bsph}(L^p) = \Theta(\mathsf{bcd}(L^p)) = \Theta(\log n)$	This paper
L^∞	$\mathsf{bsph}(L^\infty) = \mathsf{bcd}(L^\infty) = 2\log_2 n$	[25]

For an overview on the known bounds on bsph and bcd (including the results in this paper), please see Table 1.

In the full version of the paper, we give an alternate proof of the linear lower bound on $\mathsf{bsph}(L^2)$ using spectral analysis similar to that in [9]. While our lower bound is slightly weaker than the best known bounds [11, 23], our arguments require no heavy machinery and thus are arguably simpler than the previous works [11, 23, 9].

Alman and Williams [3] showed the subquadratic-time hardness for BCP in L^p-metrics, for all $p \in \mathbb{R}_{\geq 1} \cup \{0\}$, under the *Orthogonal Vector Hypothesis* (OVH). From Theorem 1 and the connection between BCP and Closest Pair described in subsection 1.1.1, we have the following hardness of Closest Pair.

▶ **Theorem 3.** *Let $p > 2$. For any $\varepsilon > 0$ and $d = \omega(\log n)$, the closest pair problem in the d-dimensional L^p-metric admits no $(n^{2-\varepsilon})$-time algorithm unless the Orthogonal Vectors Hypothesis is false.*

We remark here that showing conditional hardness for Closest Pair in the L^p metric for $p \leq 2$ remains an outstanding open problem[4]. Recently, Rubinstein [26] showed that the subquadratic-time hardness holds even for approximating BCP: Assuming OVH, for every $p \in \mathbb{R}_{\geq 1} \cup \{0\}$ and every $\varepsilon > 0$, there is a constant $\gamma(\varepsilon, p) > 0$ such that there is no $(1+\gamma)$-approximation algorithm running in time $O(n^{2-\varepsilon})$ for BCP in the L^p-metric. By using the connection between BCP and Closest Pair described in subsection 1.1.1 and the bounds in Theorem 1 (to be precise we need the efficient construction with appropriate gap as given by Theorem 17), the hardness of approximation result can be extended to Closest Pair for L^p metrics where $p > 2$.

▶ **Theorem 4.** *Let $p > 2$. For every $\varepsilon > 0$ and $d = \omega(\log n)$, there exists a constant $\gamma = \gamma(p, \varepsilon) > 0$ such that the closest pair problem in the d-dimensional L^p-metric admits no $(n^{2-\varepsilon})$-time $(1+\gamma)$-approximation algorithm unless the Orthogonal Vectors Hypothesis is false.*

We remark that the hardness for the case of the L^∞-metric does not follow (at least directly) from [3] or [26]. For independent interest, we show the subquadratic-time hardness of BCP and Closest Pair in the L^∞-metric.

▶ **Theorem 5.** *For any $\varepsilon > 0$ and $d = \omega(\log n)$, the closest pair problem in the d-dimensional L^∞-metric admits no $(n^{2-\varepsilon})$-time $(2 - o(1))$-approximation algorithm unless the Orthogonal Vectors Hypothesis is false.*

[4] The subquadratic-time hardness of Closest Pair in the L^p-metric for $p \in \mathbb{R}_{\geq 1} \cup \{0\}$ was claimed in [1] but later retracted [2].

We note that the lower bounds on bsph act as barriers for gadget reductions from BCP to Closest Pair. This partially explains why there has been no progress in showing conditional hardness for Closest Pair in the Euclidean metric for $d = \omega(\log n)$ dimensions (as $\mathsf{bsph}(L^2) = \Omega(n)$). In addition, Rubinstein noted in [26] that one obstacle in proving inapproximability results for Closest Pair is due to the triangle inequality – any two point-sets A and B in *any metric space* cannot have distinct points $a, a' \in A$ and $b \in B$ such that $\|a - a'\| > 2 \cdot \max\{\|a - b\|, \|a' - b\|\}$ (as it would violate the triangle inequality). This rules out the possibility of obtaining the conditional hardness for 2-approximating Closest Pair for any metric via simple gadget reductions. We note that the inapproximability factor of Theorem 5 matches the triangle inequality barrier (for the L^∞ metric).

1.2 Related works

While our paper studies sphericity and contact-dimension of the complete bipartite graph, determining the contact-dimension of a complete graph in L^p-metrics has also been extensively studied in the notion of *equilateral dimension*. To be precise, the equilateral dimension of a metric Δ which is the maximum number of equidistant points that can be packed in Δ. An interesting connection is in the case of the L^1-metric, for which we are unable to establish a strong lower bound for $\mathsf{bsph}(L^1)$. The equilateral dimension of L^1 is known to be at least $2d$, and this bound is believed to be tight [14]. This is a notorious open problem known as *Kusner's conjecture*, which is confirmed for $d = 2, 3, 4$ [5, 20], and the best upper bound for $d \geq 5$ is $O(d \log d)$ by Alon and Pavel [4]. If Kusner's conjecture is true for all d, then $\mathsf{sph}_1(K_n) = n/2$.

The complexity of Closest Pair has been a subject of study for many decades. There have been a series of developments on Closest Pair in the Euclidean space (see, e.g., [7, 15, 19, 27, 8]), which culminates in a deterministic $O(2^{O(d)} n \log n)$-time algorithm [8] and a randomized $O(2^{O(d)} n)$-time algorithm [24, 19]. For low (i.e., constant) dimensions, these algorithms are tight as the matching lower bound of $\Omega(n \log n)$ was shown by Ben-Or [6] and Yao [33] for the *algebraic decision tree* model, thus settling the complexity of Closest Pair in low dimensions. For high dimensions (i.e., $d = \omega(\log n)$), there is no known algorithm that runs in time significantly better than a trivial $O(n^2 d)$-time algorithm for general d except for the case that $d \geq \Omega(n)$ whereas there are subcubic-time algorithms in L^1 and L^∞ metrics [12, 16].

In the last few years, there has been a lot of progress in our understanding of BCP, Closest Pair, and related problems. Alman and Williams [3] showed subquadratic time hardness for BCP in $d = \omega(\log n)$ dimensions under OVH in the L^p metric for every $p \in \mathbb{R}_{\geq 1} \cup \{0\}$. Williams [30] extended the result of [3] and showed the above subquadratic-time hardness for BCP even for dimensions $d = \omega((\log \log n)^2)$ under OVH. In a recent breakthrough on hardness of approximation in P, Abboud et al. [2] showed the subquadratic-time hardness for approximating the Bichromatic Maximum Inner Product problem under OVH in the L^p metric for every $p \in \mathbb{R}_{\geq 1} \cup \{0\}$, and the result holds for almost polynomial approximation factors. More recently, building upon the ideas in [2], Rubinstein [26] showed under OVH the inapproximablility of BCP for every L^p-metric for $p \in \mathbb{R}_{\geq 1} \cup \{0\}$.

2 Preliminaries

We use the following standard terminologies and notations.

Distance measures. For any vector x in $\mathbb{R}^d$, we denote by $\|x\|_p$ the L^p-norm of x and is equal to $\left(\sum_{i=1}^{d} |x_i|^p\right)^{1/p}$. The L^∞-norm of x is denoted by $||x||_\infty = \max_{i\in[d]} \{|x_i|\}$, and the L^0-norm of x is denoted by $||x||_0 = |\{x_i \neq 0 : i \in [d]\}|$, i.e., the number of non-zero coordinates of x. These norms define distance measures in $\mathbb{R}^d$. The distance of two points x and y w.r.t. the L^p-norm, say *L^p-distance*, is thus $||x - y||_p$. The distance measures that are well studied in literature are the *Hamming distance* L^0-norm, the *Rectilinear distance* L^1-norm, the *Euclidean distance* L^2-norm, the *Chebyshev distance* (a.k.a, *Maximum-norm*) L^∞-norm.

Problems. Here we give formal definitions of Closest Pair and BCP. In Closest Pair, we are given a collection of points $P \subseteq \mathbb{R}^d$ in a d-dimensional L^p-metric, and the goal is find a pair of distinct points $a, b \in P$ that minimizes $\|u - v\|_p$. In BCP, the input point-set is partitioned into two color classes (the collections of red and blue points) A and B, and the goal is find a pair of points $u \in A$ and $v \in B$ that minimizes $\|u - v\|_p$.

Fine-grained complexity and conditional hardness. Conditional hardness is the current trend in proving running-time lower bounds for polynomial-time solvable problems. This has now developed into the area of *Fine-Grained Complexity*. Please see, e.g., [31, 32] and references therein.

The *Orthogonal Vectors Hypothesis* (OVH) is a popular complexity theoretic assumption in Fine-Grained Complexity. OVH states that in the Word RAM model with $O(\log n)$ bit words, any algorithm requires $n^{2-o(1)}$ time in expectation to determine whether collections of vectors $A, B \subseteq \{0,1\}^d$ with $|A| = |B| = n/2$ and $d = \omega(\log n)$ contain an orthogonal pair $u \in A$ and $v \in B$ (i.e., $\sum_{i=1}^{d} u_i \cdot v_i = 0$).

Another popular conjecture is the *Strong Exponential-Time Hypothesis* for SAT (SETH), which states that, for every $\varepsilon > 0$, there exists an integer k_ε such that k_ε-SAT on n variables cannot be solved in $O(2^{(1-\varepsilon)n})$-time. Williams showed that SETH implies OVH [29].

3 Representing biclique in L^1

In this section, we discuss the case of the L^1-metric. As discussed in the introduction, this is the only case where we are unable to prove neither strong lower bound nor linear upper bound. A weak lower bound $\mathsf{bsph}(L^1) \geq \Omega(\log n)$ follows from the proof for the L^p-metric with $p > 2$ in Section 6.1 (Theorem 16), and a quadratic upper bound $\mathsf{bcd}(L^1) \leq n^2$ follows from the proof for the L^0-metric in Section 4.2 (Corollary 12). However, we cannot prove any upper bound smaller than $\Omega(n^2)$ or any lower bound larger than $O(\log n)$. Hence, we study an average case relaxation of the question.

We show in Theorem 2 that there is no distribution whose expected distances simulate a polar-pair of point-sets in the L^1-metric. Consequently, even though we could not prove the biclique sphericity lower bound for the L^1-metric, we are able to refute an existence of a geometric representation with *large gap* for any dimension as shown in Corollary 7. (A similar result was shown in [10] for the L^2-metric.)

▶ **Definition 6** (L^1-distribution)**.** For any $d > 0$, let X, Y be two random variables taking values from $\mathbb{R}^d$. An L^1-distribution is constructed by X, Y if the following holds.

$$\begin{aligned} \mathop{\mathbb{E}}_{x_1,x_2\in_R X} [\|x_1 - x_2\|_1] &> \mathop{\mathbb{E}}_{x_1\in_R X,\, y_1\in_R Y} [\|x_1 - y_1\|_1] , \\ \mathop{\mathbb{E}}_{y_1,y_2\in_R Y} [\|y_1 - y_2\|_1] &> \mathop{\mathbb{E}}_{x_1\in_R X,\, y_1\in_R Y} [\|x_1 - y_1\|_1] . \end{aligned} \tag{6}$$

▶ **Theorem 2** (Restated). *For any two finite-supported random variables X, Y that are taking values from $\mathbb{R}^d$, there is no L^1-distribution.*

Proof. Assume towards a contradiction that there exist two finite-supported random variables X, Y that are taking values in $\mathbb{R}^d$ and that are satisfying Eq. 6 of Definition 6. Given a vector $x \in \mathbb{R}^d$, we denote by $x(i)$ the value of the i-th coordinate of x. Hence the following inequalities hold,

$$\begin{aligned}
0 &> \mathop{\mathbb{E}}_{x_1 \in_R X,\, y_1 \in_R Y} \left[\|x_1 - y_1\|_1\right] - \mathop{\mathbb{E}}_{x_1, x_2 \in_R X} \left[\|x_1 - x_2\|_1\right] \\
&= \mathop{\mathbb{E}}_{x_1, x_2 \in_R X,\, y_1 \in_R Y} \left[\|x_1 - y_1\|_1 - \|x_1 - x_2\|_1\right] \\
&= \frac{1}{d} \cdot \mathop{\mathbb{E}}_{x_1, x_2 \in_R X,\, y_1 \in_R Y} \left[\mathop{\mathbb{E}}_{i \in_R [1 \ldots d]} \left[|x_1(i) - y_1(i)| - |x_1(i) - x_2(i)|\right]\right] \\
&= \frac{1}{d} \cdot \mathop{\mathbb{E}}_{i \in_R [1 \ldots d]} \left[\mathop{\mathbb{E}}_{x_1, x_2 \in_R X,\, y_1 \in_R Y} \left[|x_1(i) - y_1(i)| - |x_1(i) - x_2(i)|\right]\right] .
\end{aligned}$$

Thus for some $i^\star \in [d]$ the following holds,

$$0 > \mathop{\mathbb{E}}_{x_1, x_2 \in_R X,\, y_1 \in_R Y} \left[|x_1(i^\star) - y_1(i^\star)| - |x_1(i^\star) - x_2(i^\star)|\right] . \tag{7}$$

Fix $i^\star \in [d]$ satisfying the above inequality. For the sake of clarity, we assume that the random variables X, Y are taking values in $\mathbb{R}$ (i.e., projection on the $i^{\star\text{th}}$ coordinate). We can assume that the size of $\mathsf{supp}(X) \cup \mathsf{supp}(Y)$ is greater than 1 because, if $\mathsf{supp}(X) \cup \mathsf{supp}(Y)$ contains a single point then $\mathop{\mathbb{E}}_{x_1 \in_R X,\, y_1 \in_R Y} \left[\|x_1 - y_1\|_1\right] = \mathop{\mathbb{E}}_{x_1, x_2 \in_R X} \left[\|x_1 - x_2\|_1\right] = 0$, contradicting Eq. 7. Let $\mathsf{supp}(X) \cup \mathsf{supp}(Y)$ contains $t \geq 2$ points. We prove by induction on t, that there are no X, Y over $\mathbb{R}$ satisfying Eq. 7. The base case is when $t = 2$. By Eq. 7, there exists 3 points $\tilde{x}_1, \tilde{x}_2, \tilde{y}_1$ in $\mathbb{R}$ such that,

$$0 > \|\tilde{x}_1 - \tilde{y}_1\|_1 - \|\tilde{x}_1 - \tilde{x}_2\|_1 . \tag{8}$$

Since $\mathsf{supp}(X) \cup \mathsf{supp}(Y)$ contains exactly two points, $\tilde{x}_1, \tilde{x}_2, \tilde{y}_1$ are supported by two distinct points in $\mathbb{R}$. Hence, there are two cases, either that $x_1 = x_2$ (and $y_1 \neq x_1$) or that $x_1 \neq x_2$ (and either $\tilde{y}_1 = \tilde{x}_1$ or $\tilde{y}_1 = \tilde{x}_2$). It is easy to see that none of these cases satisfy Eq. 8, a contradiction.

Assume the induction hypothesis that there are no X, Y taking values from $\mathbb{R}$ satisfying Eq. 7 when the size of $\mathsf{supp}(X) \cup \mathsf{supp}(Y)$ is equal to $k \geq 2$. Then consider the case when $t = k + 1 \geq 3$. Sort the points in $\mathsf{supp}(X) \cup \mathsf{supp}(Y)$ by their values, and denote by s_i the value of the i-th point of $\mathsf{supp}(X) \cup \mathsf{supp}(Y)$. For the sake of simplicity, we say that we *change* the value of s_{t-1} to $\tilde{s}_{t-1}$, where $s_{t-2} \leq \tilde{s}_{t-1} \leq s_t$, if after changing its value we change the values of (at least one of) X, Y to $\tilde{X}, \tilde{Y}$ in such a way that the value of the $(t-1)$-th point (after sorting) of $\mathsf{supp}\left(\tilde{X}\right) \cup \mathsf{supp}\left(\tilde{Y}\right)$ is equal to $\tilde{s}_{t-1}$ (if $s_{t-2} = \tilde{s}_{t-1}$, then the value of the $(t-2)$-th point of $\mathsf{supp}\left(\tilde{X}\right) \cup \mathsf{supp}\left(\tilde{Y}\right)$ is equal to $\tilde{s}_{t-1}$). Define the function $f : [s_{t-2}, s_t] \to \mathbb{R}$ as follows:

$$f(x) = \mathop{\mathbb{E}}_{x_1 \in_R \tilde{X},\, y_1 \in_R \tilde{Y}} \left[\|x_1 - y_1\|_1\right] - \mathop{\mathbb{E}}_{x_1, x_2 \in_R \tilde{X}} \left[\|x_1 - x_2\|_1\right] ,$$

where $\tilde{X}, \tilde{Y}$ are obtained after changing s_{t-1} to $x \in [s_{t-2}, s_t]$. The crucial observation is that the function f is linear. Hence, either $f(s_{t-2}) \geq f(s_{t-1})$ or $f(s_t) \geq f(s_{t-1})$, and we can reduce the size of $\mathsf{supp}(X) \cup \mathsf{supp}(Y)$ by 1. However, this contradicts our induction hypothesis. ◀

The following corollary refutes the existence of a polar-pair of point-sets with large gap in any dimension. The proof follows from Theorem 2 and is given in the full version of the paper.

▶ **Corollary 7** (No Polar-Pair of Point-Sets in L^1 with Large Gap). *For any $\alpha > 0$, there exist no subsets $A, B \subseteq \mathbb{R}^d$ of $n/2$ vectors with $d < n/2$ such that*

- *For any u, v both in A, or both in B, $\|u - v\|_1 \geq \frac{1}{1-2/n} \cdot \alpha$.*
- *For any $u \in A$ and $v \in B$, $\|u - v\|_1 < \alpha$.*

We can show similar results that there are no polar-pairs of point-sets with large gap in the L^0 and L^2 metrics. The case of the L^0-metric follows directly from Theorem 2 when the alphabet set is $\{0, 1\}$. (Please also see Lemma 9 for an alternate proof.) The case of the L^2-metric follows from the fact that $\mathsf{bsph}(L^2) = \Omega(n)$ [11, 23] and that we can reduce the dimension of a polar-pairs of point-sets with constant gap to $O(\log n)$ using dimension reduction [17].

4 Geometric representation of biclique in L^0

In this section, we prove a lower bound on $\mathsf{bsph}(L^0)$ and an upper bound on $\mathsf{bcd}(L^0)$. We start by providing a real-to-binary reduction below. Then we proceed to prove the lower bound on $\mathsf{bsph}(L^0)$ in Section 4.1 and then the upper bounds on $\mathsf{bcd}(L^0)$ in Section 4.2.

First we state the following (trivial) lemma, which allows mapping from vectors in $\mathbb{R}^d$ to zero-one vectors. The proof of the lemma can be found in the full version of the paper.

▶ **Lemma 8** (Real to Binary Reduction). *Let $S \subseteq \mathbb{R}$ be a finite set of real numbers. Then there exists a transformation $\phi : S^d \to \{0, 1\}^{d|S|}$ such that, for any $x, y \in S^d$,*

$$\|x - y\|_0 = \frac{1}{2} \cdot \|\phi(x) - \phi(y)\|_0$$

4.1 Lower bound on the biclique-sphericity

Now we will show that $\mathsf{bsph}(L^0) \geq n$. Our proof requires the following lemma, which rules out a randomized algorithm that generates a polar-pair of point-sets.

▶ **Lemma 9** (No Distribution for L^0). *For any $\alpha > \beta \geq 0$, regardless of dimension, there exist no distributions $\mathcal{A}$ and $\mathcal{B}$ of points in $\mathbb{R}^d$ with finite supports such that*

- $\mathbb{E}_{x,x' \in_R \mathcal{A}}[\|x - x'\|_0] \geq \alpha$.
- $\mathbb{E}_{y,y' \in_R \mathcal{B}}[\|y - y'\|_0] \geq \alpha$.
- $\mathbb{E}_{x \in_R \mathcal{A}, y \in_R \mathcal{B}}[\|x - y\|_0] \leq \beta$.

Proof. We prove by contradiction. Assume to a contrary that such distributions exist. Then

$$\mathbb{E}_{x,x' \in_R \mathcal{A}}[\|x - x'\|_0] + \mathbb{E}_{y,y' \in_R \mathcal{B}}[\|y - y'\|_0] - 2\mathbb{E}_{x \in_R \mathcal{A}, y \in_R \mathcal{B}}[\|x - y\|_0] > 0. \tag{9}$$

Let A and B be supports of $\mathcal{A}$ and $\mathcal{B}$, respectively. By Lemma 8, we may assume that vectors in A and B are binary vectors. Observe that each coordinate of vectors in A and B contribute to the expectations independently. In particular, Eq. (9) can be written as

$$2\sum_i \rho^A_{0,i}\rho^A_{1,i} + 2\sum_i \rho^B_{0,i}\rho^B_{1,i} + 2\sum_i \left(\rho^A_{0,i}\rho^B_{1,i} + \rho^B_{0,i}\rho^A_{1,i}\right) > 0 \tag{10}$$

where $\rho^A_{0,i}$, $\rho^A_{1,i}$, $\rho^B_{0,i}$ and $\rho^B_{1,i}$ are the probability that the i-th coordinate of $x \in A$ (resp., $y \in B$) is 0 (resp., 1). Thus, to show a contradiction, it is sufficient to consider the coordinate

which contributes the most to the summation in Eq. (10). The contribution of this coordinate to the summation is

$$2\rho_0^A\rho_1^A + 2\rho_0^B\rho_1^B - 2(\rho_0^A\rho_1^B + \rho_1^A\rho_0^B) = 2(\rho_0^A - \rho_0^B)(\rho_1^A - \rho_1^B) \tag{11}$$

Since $\rho_0^A + \rho_1^A = 1$ and $\rho_0^B + \rho_1^B = 1$, the summation in Eq.(11) can be non-negative only if $\rho_0^A = \rho_0^B$ and $\rho_1^A = \rho_1^B$. But, then this implies that the summation in Eq.(11) is zero. We have a contradiction since this coordinate contributes the most to the summation in Eq. (10) which we assume to be positive. ◀

The next Theorem shows that $\mathsf{bsph}(L^0) \geq n$.

▶ **Theorem 10** (Lower Bound for L^0 with Arbitrary Alphabet). *For any integers $\alpha > \beta \geq 0$ and $n > 0$, there exist no subsets $A, B \subseteq \mathbb{R}^d$ of n vectors with $d < n$ such that*

- *For any $a, a' \in A$, $\|a - a'\|_0 \geq \alpha$.*
- *For any $b, b' \in B$, $\|b - b'\|_0 \geq \alpha$.*
- *For any $a \in A$ and $b \in B$, $\|a - b\|_0 \leq \beta$.*

Proof. Suppose for a contradiction that such subsets A and B exist with $d < n$. We build uniform distributions $\mathcal{A}$ and $\mathcal{B}$ by uniformly at random picking a vector in A and B, respectively. Then it is easy to see that the expected value of inner distance is

$$\mathbb{E}_{x,x' \in_R \mathcal{A}}[\|x - x'\|_0] \geq \alpha - \frac{\alpha}{n}$$

The intra distance of B is similar. We know that $\alpha - \beta \geq 1$ because they are integers and so are L^0-distances. But, then if $\alpha < n$, we would have distributions that contradict Lemma 9. Note that α and β are at most d (dimension). Therefore, we conclude that $d \geq n$. ◀

4.2 Upper bound on the biclique contact-dimension

Now we show that $\mathsf{bcd}(L^0) \leq n$.

▶ **Theorem 11** (Upper Bound for L^0 with Arbitrary Alphabet). *For any integer $n > 0$ and $d = n$, there exist subsets $A, B \subseteq \mathbb{R}^d$ each with n vectors such that*

- *For any $a, a' \in A$, $\|a - a'\|_0 = d$.*
- *For any $b, b' \in B$, $\|b - b'\|_0 = d$.*
- *For any $a \in A$ and $b \in B$, $\|a - b\|_0 = d - 1$.*

Proof. First we construct a set of vectors A. For $i = 1, 2, \ldots, n$, we define the i-th vector a of A so that a is an all-i vector. That is, $a = (i, i, \ldots, i)$. Next we construct a set of vectors B. The first vector of B is $(1, 2, \ldots, n)$. Then the $(i+1)$-th vector of B is the left rotation of the i-th vector. Thus, the i-th vector of B is $b = (i, i+1, \ldots, n, 1, 2, \ldots, i-1)$.

It can be seen that the L^0-distance between any two vectors from the same set is d because all the coordinates are different. Any vectors from different set, say $a \in A$ and $b \in B$, must have at least one common coordinate. Thus, their L^0-distance is $d - 1$. This proves the lemma. ◀

Below is the upper bound for zero-one vectors, which is a corollary of Theorem 11. The proof can be found in the full version of the paper.

▶ **Corollary 12** (Upper Bound for L^0 with Binary Vectors). *For any integer $n > 0$ and $d = n^2$, there exist subsets $A, B \subseteq \mathbb{R}^d$ each with n vectors such that*

- *For any $a, a' \in A$, $\|a - a'\|_0 = n$.*
- *For any $b, b' \in B$, $\|b - b'\|_0 = n$.*
- *For any $a \in A$ and $b \in B$, $\|a - b\|_0 = n - 1$.*

5 Geometric representation of biclique in L^p for $p \in (1, 2)$

In this section, we prove the upper bound on $\mathsf{bcd}(L^p)$ for $p \in (1, 2)$. We are unable to show any lower bound for these L_p-metrics except for the lower bound of $\Omega(\log n)$ obtained from the ϵ-net lower bound in Theorem 16 (which will be proven in the next Section).

▶ **Theorem 13** (Upper Bound for L^p with $1 < p < 2$)**.** *For every $1 < p < 2$ and for all integers $n \geq 1$, there exist two sets $A, B \subseteq \mathbb{R}^{2n}$ each of cardinality n such that the following holds:*

1. *For every distinct points $u, v \in A$, $\|u - v\|_p = 2^{1/p}$.*
2. *For every distinct points $u, v \in B$, $\|u - v\|_p = 2^{1/p}$.*
3. *For every points $u \in A$ and $v \in B$, $\|u - v\|_p < 2^{1/p}$.*

Proof. We will construct point-sets as claimed in the theorem for given p and n. Let α be a parameter depending on p and n, which will be set later. For each $i \in [n]$, we create a point $a \in A$ by setting

$$a_j = \begin{cases} 0 & \text{if } 1 \leq j \leq n \text{ and } j \neq i \\ 1 & \text{if } 1 \leq j \leq n \text{ and } i = j \\ \alpha & \text{if } n+1 \leq j \leq 2n \end{cases}$$

Similarly, for each $i \in [n]$, we create a point $b \in B$ by setting

$$b_j = \begin{cases} \alpha & \text{if } 1 \leq j \leq n \\ 0 & \text{if } n+1 \leq j \leq 2n \text{ and } j \neq n+i \\ 1 & \text{if } n+1 \leq j \leq 2n \text{ and } j = n+i \end{cases}$$

By construction, for every pair of points u, v both in A or both in B, their L^p-distance is $\|u - v\|_p = 2^{1/p}$, and for every pair of points from different sets, say $u \in A$ and $v \in B$, their L^p-distance is

$$\|u - v\|_p = 2^{1/p} \cdot ((1-\alpha)^p + (n-1) \cdot \alpha^p)^{1/p} \leq 2^{1/p} \cdot ((1-\alpha)^p + n \cdot \alpha^p)^{1/p} \tag{12}$$

Now let us choose $\alpha > n^{-1/(p-1)}$, and consider the term $(1-\alpha)^p + n \cdot \alpha^p$ in Eq. (12). Observe that $\alpha < n \cdot \alpha^p$ for $1 < p < 2$. Define a function $f(x) = (1-\alpha)^x + n \cdot \alpha^x$. We know that $f(x)$ is less than 1 as x goes from ∞ to 1 (i.e., $\lim_{x \to 1^+}((1-\alpha)^x + n \cdot \alpha^x) < 1$). Moreover, $f(x)$ is decreasing for $0 < \alpha < 1$, which means that $f(p) < 1$. Consequently, $\|u - v\|_p < 2^{1/p}$, and the theorem follows.

To finish the proof, we will show that $f(x)$ is decreasing for $x > 1$ provided that $0 < \alpha < 1$. It suffices to show that $f'(x) < 0$ for all values of x.

$$f'(x) = \frac{\partial}{\partial x}\left((1-\alpha)^x + n \cdot \alpha^x\right) = (1-\alpha)^x \ln(1-\alpha) + n \cdot \alpha^x \ln(\alpha) < 0.$$

The last inequality follows from the fact $\ln(x) < 0$ for $0 < x < 1$ and that $0 < \alpha, 1-\alpha < 1$. ◀

6 Geometric representation of biclique in L^p for $p > 2$

In this section, we show the lower bound on $\mathsf{bsph}(L^p)$ and an upper bound on $\mathsf{bcd}(L^p)$ for $p > 2$. Both bounds are logarithmic. The latter upper bound is constructive and efficient (in the sense that the polar-pair of point-sets can be constructed in $\widetilde{O}(n)$-time). This implies the subquadratic-time equivalence between Closest Pair and BCP.

6.1 Lower bound on the biclique sphericity

Now we show the lower bound on the biclique sphericity of a complete bipartite graph in L^p-metrics with $p > 2$. In fact, we prove the lower bound for the case of a star graph on n vertices, denoted by S_n, and then use the fact that $\mathsf{bsph}(H) \leq \mathsf{bsph}(G)$ for all induced subgraph H of G (i.e., $\mathsf{bsph}(K_{n/2,n/2}, L^p) \geq \mathsf{bsph}(S_{n/2}, L^p)$).

In short, we show in Lemma 16 that $O(\log n)$ is the maximum number of L^p-balls of radius $1/2$ that we can pack in an L^p-ball of radius one so that no two of them intersect or touch each other. This upper bounds, in turn, implies the lower bound on the dimension. We proceed with the proof by volume arguments, which are commonly used in proving the minimum number of points in an ϵ-*net* that are sufficient to cover all the points in a sphere.

▶ **Definition 14** (ϵ-net)**.** The unit L^p-ball in $\mathbb{R}^d$ centered at o is denoted by

$$\mathbb{B}\left(L_p^d, o\right) = \left\{x \in \mathbb{R}^d \mid \|x - o\|_p \leq 1\right\}.$$

For brevity, we write $\mathbb{B}\left(L_p^d\right)$ to mean $\mathbb{B}\left(L_p^d, o\right)$. Let (X, d) be a metric space and let S be a subset of X and ϵ be a constant greater than 0. A subset N_ϵ of X is called an **ϵ-net** of S under d if for every point $x \in S$ it holds for some point $y \in N_\epsilon$ that $d(x, y) \leq \epsilon$.

The following lemma is well known in literature (see, e.g., [28]). For the sake of completeness, we provide a proof in the full version of the paper.

▶ **Lemma 15.** *There exists an ϵ-net for $\mathbb{B}\left(L_p^d\right)$ under the L^p-metric of cardinality $\left(1 + \frac{2}{\epsilon}\right)^d$.*

▶ **Theorem 16.** *For every $N, d \in \mathbb{N}$, for $p \geq 1$, and for any two sets $A, B \subseteq \mathbb{R}^d$, each of cardinality N, suppose the following holds for some non-negative real numbers α and β with $\alpha > \beta$.*

1. *For every u and v both in A, $\|u - v\|_p > \alpha$.*
2. *For every u and v both in B, $\|u - v\|_p > \alpha$.*
3. *For every u in A and v in B, $\|u - v\|_p \leq \beta$.*

Then the dimension d must be at least $\log_5(N)$.

Proof. Scale and translate the sets A, B in such a way that $\beta = 1$ and that $\vec{0} \in B$. It follows that $A \subseteq \mathbb{B}\left(L_p^d\right)$. By Lemma 15, we can fix a 1/2-net $N_{1/2}$ for $\mathbb{B}\left(L_p^d\right)$ of size 5^d. Note that, for every $x \in N_{1/2}$, the ball $1/2 \cdot \mathbb{B}\left(L_p^d, x\right)$ contains at most one point from A. Note also that $N_{1/2}$ covers $\mathbb{B}\left(L_p^d\right)$. Thus, $|A| \leq 5^d$ which implies that $d \geq \log_5(N)$. ◀

6.2 Upper bound on the biclique contact-dimension

We first give a simple randomized construction that gives a logarithmic upper bound on the biclique contact-dimension of L^p. The construction is simple. We uniformly at random take a subset A of n vectors from $\{-1, 1\}^{d/2} \times \{0\}^{d/2}$ and a subset B of n vectors from $\{0\}^{d/2} \times \{-1, 1\}^{d/2}$. Observe that, for any $p > 2$, the L^p-distance of any pair of vectors $u \in A$ and $v \in B$ is exactly d while the *expected distance* between the inner pair $u, u' \in A$ (resp., $v, v' \in B$) is strictly larger than d. Thus, if we choose d to be sufficiently large, e.g., $d \geq 10 \ln n$, then we can show by a standard concentration bound (e.g., Chernoff's bound) that the probability that the inner-pair distance is strictly larger than d is at least $1 - 1/n^3$. Applying the union bound over all inner-pairs, we have that the d-neighborhood graph of $A \cup B$ is a bipartite complete graph with high probability. Moreover, the distances between any crossing pairs $u \in A$ and $v \in B$ are the same for all pairs. This shows the upper bound for the contact-dimension of a biclique in the L^p-metric for $p > 2$.

The above gives a simple proof of the upper bound on the biclique contact-dimension of the L^p-metric. Moreover, it shows a randomized construction of the polar-pair in the $O(\log n)$-dimensional L^p-metric, for $p > 2$, thus implying that Closest Pair and BCP are equivalent for these L^p-metrics.

For algorithmic purposes, we provide a deterministic construction below using appropriate binary codes.

▶ **Theorem 17.** *For any $p > 2$, let $\zeta = 2^{p-3}$. There exist two sets $|A| = |B| = n$ of vectors in $\mathbb{R}^d$, where $d = 2\alpha \log_2 n$, for some constant $\alpha \geq 1$, such that the following holds.*

1. *For all $u, u' \in A$, $\|u - u'\|_p > ((\zeta + 1/2)d)^{1/p}$.*
2. *For all $v, v' \in B$, $\|v - v'\|_p > ((\zeta + 1/2)d)^{1/p}$.*
3. *For all $u \in A$, $v \in B$, $\|u - v\|_p = d^{1/p}$.*

Moreover, there exists a deterministic algorithm that outputs A and B in time $\widetilde{O}(n)$.

Proof. In literature, we note that for any constant $\delta > 0$, there is an explicit binary code of (some) constant *relative rate* and *relative distance* at least $\frac{1}{2} - \delta$ and the entire code can be listed in quasilinear time with respect to the size of the code (see Appendix E.1.2.5 from [13], or Justesen codes [18]). To be more specific, we can construct in $O(n \log^{O(1)} n)$-time a set $C \subseteq \{-1, 1\}^{d'}$ such that (1) $|C| = n$, (2) $d' = d/2 = \alpha \log_2 n$ for some constant $\alpha \geq 1$ and (3) for every two vectors $x, y \in C$, x and y differ on at least $\left(\frac{1}{2} - \delta\right) d'$ coordinates, for some constant $\delta \in \left(0, \frac{1}{4} - \frac{1}{2^p}\right)$.

We construct the sets A and B as subsets of $\{-1, 0, 1\}^d$. For every $i \in [n]$, the i^{th} point of A is given by the concatenation of the i^{th} point of C with $0^{d'}$. Similarly, the i^{th} point of B is given by the concatenation of $0^{d'}$ with the i^{th} point of C (note the reversal in the order of the concatenation). In particular, points in A and B are of the form $(x_i, \vec{0})$ and $(\vec{0}, x_i)$, respectively, where x_i is the i^{th} point in C and $\vec{0}$ is the zero-vector of length $\alpha \log_2 n$.

First, consider any two points in the same set, say $u, u' \in A$ (resp., $v, v' \in B$). We have from the distance of C that on at least $\left(\frac{1}{2} - \delta\right) d'$ coordinates the two points differ by 2, thus implying that their L^p-distance is at least

$$\left(\left(\frac{1}{2} - \delta\right) d' 2^p\right)^{1/p} > \left(\left(\frac{1}{4} + \frac{1}{2^p}\right) d' 2^p\right)^{1/p} = \left(\left(2^{p-3} + \frac{1}{2}\right) d\right)^{1/p}.$$

This proves the first two items of the theorem. Next we prove the third item. Consider any two points from different sets, say $u \in A$ and $v \in B$. It is easy to see from the construction that u and v differ in every coordinate by exactly 1. Thus, the L^p-distance between any two points from different set is exactly

$$(2d')^{1/p} = d^{1/p}.$$ ◀

7 Fine-grained complexity of CLOSEST PAIR in L^∞

In this section, we prove the quadratic-time hardness of Closest Pair in the L^∞-metric. Our reduction is from the *Orthogonal Vectors* problem (OV), which we phrase it as follows. Given a pair of collections of vectors $U, W \subseteq \{0, 1\}^d$, the goal is to find a pair of vectors $u \in U$ and $w \in W$ such that $(u_i, w_i) \in \{(0, 0), (0, 1), (1, 0)\}$ for all $i \in [d]$. Throughout, we denote by n the total number of vectors in U and W.

Let $U, W \subseteq \{0, 1\}^d$ be an instance of OV. We may assume that U and W have no duplicates. Otherwise, we may sort vectors in U (resp., W) in lexicographic order and then sequentially remove duplicates; this preprocessing takes $O(dn \log n)$-time.

We construct a pair of sets $A, B \subseteq \mathbb{R}^d$ of BCP from U, W as follows. For each vector $u \in U$ (resp., $w \in W$), we create a point $a \in A$ (resp., $b \in B$) such that

$$a_j = \begin{cases} 0 & \text{if } u_j = 0, \\ 2 & \text{if } u_j = 1. \end{cases}$$

$$b_j = \begin{cases} 1 & \text{if } w_j = 0, \\ -1 & \text{if } w_j = 1. \end{cases}$$

Observe that, for any vectors $a \in A$ and $b \in B$, $|a_j - b_j| = 3$ only if $u_j = w_j = 1$; otherwise, $|a_j - b_j| = 1$. It can be seen that $\|a - b\|_p = d$ if and only if their corresponding vectors $u \in U$ and $w \in W$ are orthogonal. Thus, this gives an alternate proof for the quadratic-time hardness of BCP under OVH.

Here we show that the reduction above rules out both exact and $(2 - o(1))$-approximation algorithm for Closest Pair in L^∞ that runs in subquadratic-time (unless OVH is false). That is, we prove Theorem 5, which follows from the theorem below whose proof is in the full version of the paper. In short, given an instance (U, W) of the OV problem, the instance of Closest Pair that is constructed in the reduction in simply $A \cup B$.

▶ **Theorem 18.** *Assuming OVH, for any $\varepsilon > 0$ and $d = \omega(\log n)$, there is no $O\left(n^{2-\varepsilon}\right)$-time algorithm that, given a point-set $P \subseteq \mathbb{R}^d$, distinguishes between the following two cases:*

- *There exists a pair of vectors in P with L^∞-distance one.*
- *Every pair of vectors in P has L^∞-distance two.*

In particular, approximating Closest Pair *in the L^∞-metric to within a factor of two is at least as hard as solving the Orthogonal Vectors problem.*

8 Conclusion and discussion

We have studied the sphericity and contact dimension of the complete bipartite graph in various metrics. We have proved lower and upper bounds on these measures for some metrics. However, biclique sphericity and biclique contact dimension in the L^1-metric remains poorly understood as we are unable to show any strong upper or lower bounds. However, we believe that both L^1 and L^2 metrics have linear upper and lower bounds. To be precise, we raise the following conjecture:

▶ **Conjecture 19** (L^1-Biclique Sphericity Conjecture)**.**

$$\mathsf{bsph}(L^1) = \Omega(n).$$

We have also shown conditional lower bounds for the Closest Pair problem in the L^p-metric, for all $p \in \mathbb{R}_{>2} \cup \{\infty\}$, by using polar-pair of point-sets. However, it is unlikely that our techniques could get to the regime of L^2, L^1, and L^0, which are popular metrics. An open question is thus whether there exists an alternative technique to derive a lower bound from OVH to the Closest Pair problem for these metrics. The answer might be on the positive side, i.e., there might exist an algorithm that performs well in the L^2-metric because there are more tools available, e.g., Johnson-Lindenstrauss' dimension reduction. Thus, it is possible that there exists a strongly subquadratic-time algorithm in the L^2-metric. This question remains an outstanding open problem.

References

1. Amir Abboud, Aviad Rubinstein, and R. Ryan Williams. Distributed PCP theorems for hardness of approximation in P. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 25–36, 2017. doi:10.1109/FOCS.2017.12.
2. Amir Abboud, Aviad Rubinstein, and Ryan Williams. Distributed PCP theorems for hardness of approximation in P. *CoRR*, abs/1706.06407, 2017. Preliminary version in FOCS'17. URL: http://arxiv.org/abs/1706.06407, arXiv:1706.06407.
3. Josh Alman and Ryan Williams. Probabilistic polynomials and hamming nearest neighbors. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 136–150, 2015. doi:10.1109/FOCS.2015.18.
4. Noga Alon and Pavel Pudlák. Equilateral sets in ℓ_p^n. *Geometric & Functional Analysis GAFA*, 13(3):467–482, 2003. doi:10.1007/s00039-003-0418-7.
5. Hans-Jürgen Bandelt, Victor Chepoi, and Monique Laurent. Embedding into rectilinear spaces. *Discrete & Computational Geometry*, 19(4):595–604, 1998. doi:10.1007/PL00009370.
6. Michael Ben-Or. Lower bounds for algebraic computation trees (preliminary report). In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, USA*, pages 80–86, 1983. doi:10.1145/800061.808735.
7. Jon Louis Bentley. Multidimensional divide-and-conquer. *Commun. ACM*, 23(4):214–229, 1980. doi:10.1145/358841.358850.
8. Jon Louis Bentley and Michael Ian Shamos. Divide-and-conquer in multidimensional space. In *Proceedings of the 8th Annual ACM Symposium on Theory of Computing, May 3-5, 1976, Hershey, Pennsylvania, USA*, pages 220–230, 1976. doi:10.1145/800113.803652.
9. Yonatan Bilu and Nathan Linial. Monotone maps, sphericity and bounded second eigenvalue. *J. Comb. Theory, Ser. B*, 95(2):283–299, 2005. doi:10.1016/j.jctb.2005.04.005.
10. Michel Deza and Hiroshi Maehara. A few applications of negative- type inequalities. *Graphs and Combinatorics*, 10(2-4):255–262, 1994. doi:10.1007/BF02986674.
11. Peter Frankl and Hiroshi Maehara. On the contact dimensions of graphs. *Discrete & Computational Geometry*, 3:89–96, 1988. doi:10.1007/BF02187899.
12. Omer Gold and Micha Sharir. Dominance products and faster algorithms for high-dimensional closest pair under l_∞. *CoRR*, abs/1605.08107, 2016. URL: http://arxiv.org/abs/1605.08107, arXiv:1605.08107.
13. Oded Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, New York, NY, USA, 1 edition, 2008.
14. Richard K Guy. An olla-podrida of open problems, often oddly posed. *The American Mathematical Monthly*, 90(3):196–200, 1983.
15. Klaus H. Hinrichs, Jürg Nievergelt, and Peter Schorn. Plane-sweep solves the closest pair problem elegantly. *Inf. Process. Lett.*, 26(5):255–261, 1988. doi:10.1016/0020-0190(88)90150-0.
16. Piotr Indyk, Moshe Lewenstein, Ohad Lipsky, and Ely Porat. Closest pair problems in very high dimensions. In *Automata, Languages and Programming: 31st International Colloquium, ICALP 2004, Turku, Finland, July 12-16, 2004. Proceedings*, pages 782–792, 2004. doi:10.1007/978-3-540-27836-8_66.
17. William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.
18. Jørn Justesen. Class of constructive asymptotically good algebraic codes. *IEEE Trans. Information Theory*, 18(5):652–656, 1972. doi:10.1109/TIT.1972.1054893.
19. Samir Khuller and Yossi Matias. A simple randomized sieve algorithm for the closest-pair problem. *Inf. Comput.*, 118(1):34–37, 1995. doi:10.1006/inco.1995.1049.

20 Jack H. Koolen, Monique Laurent, and Alexander Schrijver. Equilateral dimension of the rectilinear space. *Des. Codes Cryptography*, 21(1/3):149–164, 2000.

21 Hiroshi Maehara. Space graphs and sphericity. *Discrete Applied Mathematics*, 7(1):55–64, 1984.

22 Hiroshi Maehara. Contact patterns of equal nonoverlapping spheres. *Graphs and Combinatorics*, 1(1):271–282, 1985. doi:10.1007/BF02582952.

23 Hiroshi Maehara. Dispersed points and geometric embedding of complete bipartite graphs. *Discrete & Computational Geometry*, 6:57–67, 1991. doi:10.1007/BF02574674.

24 Michael O. Rabin. Probabilistic algorithms. In *Proceedings of a Symposium on New Directions and Recent Results in Algorithms and Complexity, Computer Science Department, Carnegie-Mellon University, April 7-9, 1976*, pages 21–39, 1976.

25 Fred S Roberts. On the boxicity and cubicity of a graph. *Recent Progresses in Combinatorics*, pages 301–310, 1969.

26 Aviad Rubinstein. Hardness of approximate nearest neighbor search. *CoRR*, abs/1803.00904, 2018. arXiv:1803.00904.

27 Michael Ian Shamos and Dan Hoey. Closest-point problems. In *16th Annual Symposium on Foundations of Computer Science, Berkeley, California, USA, October 13-15, 1975*, pages 151–162, 1975. doi:10.1109/SFCS.1975.8.

28 Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. *CoRR*, abs/1011.3027, 2010. URL: http://arxiv.org/abs/1011.3027, arXiv:1011.3027.

29 Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2-3):357–365, 2005. Preliminary version in ICALP'04. doi:10.1016/j.tcs.2005.09.023.

30 Ryan Williams. On the difference between closest, furthest, and orthogonal pairs: Nearly-linear vs barely-subquadratic complexity in computational geometry. *arXiv preprint arXiv:1709.05282*, 2017.

31 Virginia Vassilevska Williams. Hardness of easy problems: Basing hardness on popular conjectures such as the strong exponential time hypothesis (invited talk). In *10th International Symposium on Parameterized and Exact Computation, IPEC 2015, September 16-18, 2015, Patras, Greece*, pages 17–29, 2015. doi:10.4230/LIPIcs.IPEC.2015.17.

32 Virginia Vassilevska Williams. Fine-grained algorithms and complexity (invited talk). In *33rd Symposium on Theoretical Aspects of Computer Science, STACS 2016, February 17-20, 2016, Orléans, France*, pages 3:1–3:1, 2016. doi:10.4230/LIPIcs.STACS.2016.3.

33 Andrew Chi-Chih Yao. Lower bounds for algebraic computation trees with integer inputs. *SIAM J. Comput.*, 20(4):655–668, 1991. Preliminary version in FOCS'89. doi:10.1137/0220041.

Coordinated Motion Planning: Reconfiguring a Swarm of Labeled Robots with Bounded Stretch*

Erik D. Demaine
MIT Computer Science and Artificial Intelligence Laboratory
Cambridge, MA, USA
edemaine@mit.edu

Sándor P. Fekete
Department of Computer Science, TU Braunschweig
Mühlenpfordtstr. 23, 38106 Braunschweig, Germany
s.fekete@tu-bs.de
https://orcid.org/0000-0002-9062-4241

Phillip Keldenich[1]
Department of Computer Science, TU Braunschweig
Mühlenpfordtstr. 23, 38106 Braunschweig, Germany
p.keldenich@tu-bs.de
https://orcid.org/0000-0002-6677-5090

Christian Scheffer
Department of Computer Science, TU Braunschweig
Mühlenpfordtstr. 23, 38106 Braunschweig, Germany
c.scheffer@tu-bs.de
https://orcid.org/0000-0002-3471-2706

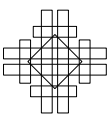
Henk Meijer
Science Department, University College Roosevelt Middelburg,
Middelburg, The Netherlands
h.meijer@ucr.nl

Abstract

We present a number of breakthroughs for coordinated motion planning, in which the objective is to reconfigure a swarm of labeled convex objects by a combination of parallel, continuous, collision-free translations into a given target arrangement. Problems of this type can be traced back to the classic work of Schwartz and Sharir (1983), who gave a method for deciding the existence of a coordinated motion for a set of disks between obstacles; their approach is polynomial in the complexity of the obstacles, but exponential in the number of disks. Despite a broad range of other non-trivial results for multi-object motion planning, previous work has largely focused on *sequential* schedules, in which one robot moves at a time, with objectives such as the number of moves; attempts to minimize the overall makespan of a coordinated *parallel* motion schedule (with many robots moving simultaneously) have defied all attempts at establishing the complexity in the absence of obstacles, as well as the existence of efficient approximation methods.

We resolve these open problems by developing a framework that provides constant-factor approximation algorithms for minimizing the execution time of a coordinated, *parallel* motion plan for a swarm of robots in the absence of obstacles, provided their arrangement entails some amount of separability. In fact, our algorithm achieves *constant stretch factor*: If all robots want to move at most d units from their respective starting positions, then the total duration of the overall schedule (and hence the distance traveled by each robot) is $O(d)$. Various extensions

* This work was partially supported by the DFG Research Unit *Controlling Concurrent Change*, funding number FOR 1800, project FE407/17-2, Conflict Resolution and Optimization.
[1] Supported by the German Research Foundation under Grant No. FE 407/17-2.



34th Symposium on Computational Geometry (SoCG 2018).
Editors: Bettina Speckmann and Csaba Tóth; Article No. 29; pp. 29:1–29:15

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

include unlabeled robots and different classes of robots. We also resolve the complexity of finding a reconfiguration plan with minimal execution time by proving that this is NP-hard, even for a grid arrangement without any stationary obstacles. On the other hand, we show that for densely packed disks that cannot be well separated, a stretch factor $\Omega(N^{1/4})$ may be required. On the positive side, we establish a stretch factor of $O(N^{1/2})$ even in this case. The intricate difficulties of computing precise optimal solutions are demonstrated by the seemingly simple case of just two disks, which is shown to be excruciatingly difficult to solve to optimality.

2012 ACM Subject Classification Theory of computation → Computational geometry, Theory of computation → Problems, reductions and completeness, Computer systems organization → Robotic control

Keywords and phrases Robot swarms, coordinated motion planning, parallel motion, makespan, bounded stretch, complexity, approximation

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.29

Related Version A full version of this paper can be found at `https://arxiv.org/abs/1801.01689` [6]. There is a video motivating, visualizing and demonstrating the concepts of this paper, reachable via `http://computational-geometry.org/SoCG-videos/socg18video/videos/74`, with an accompanying abstract at [4] in these proceedings, `http://dx.doi.org/10.4230/LIPIcs.SoCG.2018.74`.

Acknowledgements We thank anonymous reviewers of a preliminary version of the paper for helping to improve the overall presentation.

1 Introduction

Since the beginning of computational geometry, robot motion planning has been at the focus of algorithmic research. Planning the relocation of a geometric object among geometric obstacles leads to intricate scientific challenges, requiring the combination of deep geometric and mathematical insights with algorithmic techniques. With the broad and ongoing progress in robotics, the increasing importance of intelligent global planning with performance guarantees requires more sophisticated algorithmic reasoning, in particular when it comes to the higher-level task of coordinating the motion of many robots.

From the early days, multi-robot coordination has received attention from the algorithmic side. Even in the groundbreaking work by Schwartz and Sharir [20] from the 1980s, one of the challenges was coordinating the motion of *several* disk-shaped objects among obstacles. Their algorithms run in time polynomial in the complexity of the obstacles, but exponential in the number of disks. This illustrates the significant challenge of coordinating many individual robots. In addition, a growing number of applications focus primarily on robot interaction in the absence of obstacles, such as air traffic control or swarm robotics, where the goal is overall efficiency, rather than individual navigation.

With the challenges of multi-robot coordination being well known, there is still a huge demand for positive results with provable performance guarantees. In this paper, we provide significant progress in this direction, with a broad spectrum of results.

1.1 Our results

- For the problem of minimizing the total time for reconfiguring a system of labeled circular robots in a grid environment, we show that it is strongly NP-complete to compute an

optimal solution; see Theorem 1.

- We give an $\mathcal{O}(1)$-approximation for the long-standing open problems of parallel motion-planning with minimum makespan in a grid setting. This result is based on establishing an *absolute* performance guarantee: We prove that for any labeled arrangement of robots, there is always an overall schedule that gets each robot to its target destination with bounded *stretch*, i.e., within a constant factor of the largest individual distance. See Theorem 3 for the base case of grid-based configurations, which is extended later on.
- For our approach, we make use of a technique to separate planar (cyclic) flows into so-called *subflows* whose thickness can be controlled by the number of subflows, see Definition 7 and Lemma 8. This is of independent interest for the area of packet routing with bounded memory: Our Theorem 4 implies that $\mathcal{O}(D)$ steps are sufficient to route any permutation of dilation D on the grid, even with a buffer size of 1, resolving an open question by Scheideler [19] dating back to 1998.
- We extend our approach to establish constant stretch for the generalization of *colored* robot classes, for which unlabeled robots are another special case; see Theorem 13.
- We extend our results to the scenario with continuous motion and arbitrary coordinates, provided the distance between a robot's start and target positions is at least one diameter; see Theorem 15. This implies that efficient multi-robot coordination is always possible under relatively mild separability conditions; this includes non-convex robots.
- For the continuous case of N unit disks and weaker separability, we establish a lower bound of $\Omega(N^{1/4})$ and an upper bound of $\mathcal{O}(\sqrt{N})$ on the achievable stretch, see Theorem 14 and Theorem 15.

We also highlight the geometric difficulty of computing optimal trajectories even in seemingly simple cases; due to limited space, this can be found in the full version of the paper [6].

1.2 Related work

Multi-object motion planning problems have received a tremendous amount of attention from a wide spectrum of areas. Due to limited space, we focus on algorithmic work with an emphasis on geometry; see the full version of the paper [6] for a more comprehensive overview.

In the presence of obstacles, Aronov et al. [3] demonstrate that for up to three robots, a path can be constructed efficiently, if one exists. Ramanathan and Alagar [17] and Schwartz and Sharir [20] consider the case of several disk-shaped objects between polygonal obstacles. Both give algorithms for deciding reachability of a given target configuration. The algorithms run in time polynomial in the complexity of the obstacles, but exponential in the number of disks. Hopcroft et al. [11] and Hopcroft and Wilfong [12] prove that it is PSPACE-complete to decide reachability of a given target configuration, even when restricted to rectangular objects in a rectangular region. This was later strengthened by Hearn and Demaine [9, 10] to rectangles of size 1×2 and 2×1. Moreover, this problem is similar to the well-known *Rush-Hour Problem*, which was shown to be PSPACE-complete by Flake and Baum [8]. For moving disks, Spirakis and Yap [24] prove strong NP-hardness of the same problem for disks of varying size. Bereg et al. [5] and Abellanas et al. [1] consider minimizing the *number* of moves of a set of disks into a target arrangement without obstacles. These bounds were later improved by Dumitrescu and Jiang [7], who prove that the problem remains NP-hard for congruent disks even when the motion is restricted to sliding. Yu [27] provides an expected constant-factor approximation for the optimal makespan in the grid case.

On the practical side, there is a wide range of approaches for solving multi-object motion planning problems, both optimally and heuristically; for instances of limited size, SAT solvers and IP-based methods are used for discretized versions, while for larger instances, previous work resorts to heuristic solutions. Refer to the full version of the paper [6] for an overview.

In both discrete and geometric variants of the problem, the objects can be *labeled*, *colored* or *unlabeled*. In the *colored* case, the objects are partitioned into k groups and each target position can only be covered by an object with the right color. This case was recently considered by Solovey and Halperin [21], who present and evaluate a practical sampling-based algorithm. In the *unlabeled* case, the objects are indistinguishable and each target position can be covered by any object. This scenario was first considered by Kloder and Hutchinson [13], who presented a practical sampling-based algorithm. Turpin et al. [25] prove that it is possible to find a solution in polynomial time, if one exists. This solution is optimal with respect to the longest distance traveled by any one robot. However, their results only hold for disk-shaped robots under additional restrictive assumptions on the free space. For unit disks and simple polygons, Adler et al. [2] provide a polynomial-time algorithm under the additional assumption that the start and target positions have some minimal distance from each other. Under similar separability assumptions, Solovey et al. [22] provide a polynomial-time algorithm that produces a set of paths that is no longer than $\mathrm{OPT} + 4N$, where N is the number of robots. However, they do not consider the makespan, but only the total path length. On the negative side, Solovey and Halperin [23] prove that the unlabeled multiple-object motion planning problem is PSPACE-hard, even when restricted to unit square objects in a polygonal environment.

The problem of finding constructive algorithmic solutions for the problem of coordinated, parallel motion planning in the absence of obstacles (with the objective of minimizing the makespan of the overall schedule) was explicitly posed as a long-standing open problem by Overmars [16] at the 2006 Dagstuhl meeting on Robot Navigation, but can be traced back much further. It is also related to open problems from the field of routing, where it is well-known that for any given family of simple paths one can find a way to route packets along the paths such that the total number of steps required is $\mathcal{O}(C + D)$, where C is the congestion and D is the dilation of the given family of paths. However, algorithms in this context typically require that each node can store a constant number of packets. Scheideler [19] raises the question whether routing in $\mathcal{O}(C + D)$ steps is still possible if only one packet can be stored at each node. We answer a variant of this question in the case of Grid Permutation Routing. By our Theorem 4, $\mathcal{O}(D)$ steps are sufficient to route any permutation of dilation D on the grid, even with a buffer size of 1.

On the other hand, on grid graphs, the problem resembles the generalization of the 15-puzzle, for which Wilson [26] and Kornhauser et al. [14] give an efficient algorithm that decides reachability of a target configuration and provide both lower and upper bounds on the number of moves required. Ratner and Warmuth [18] prove finding a shortest solution for this puzzle remains NP-hard.

During the review period of our work, Yu [28] has independently proposed a similar approach that also achieves a constant-factor approximation in the case of a rectangular grid.

2 Preliminaries

In the grid setting of Section 3 we consider an $n_1 \times n_2$-grid $G = (V, E)$, which is dual to an $n_1 \times n_2$-rectangle P in which the considered robots are arranged. A *configuration* of P is a mapping $C : V \to \{1, \ldots, N, \perp\}$, which is injective w.r.t. the labels $\{1, \ldots, N\}$ of the

$N \leq |P|$ robots to be moved, where $\perp$ denotes the empty square. The inverse image of a robot's label ℓ is $C^{-1}(\ell)$. In the following, we consider a *start configuration* C_s and *target configuration* C_t; for $i \in \{1, \ldots, N\}$, we call $C_s^{-1}(i)$ and $C_t^{-1}(i)$ the *start* and *target position* of the robot i. Given the (minimum) Manhattan distance between each robot's start/target positions for each robot, we denote by d the maximum such distance over all robots.

A configuration $C_1 : V \to \{1, \ldots, N, \perp\}$ can be *transformed within one single transformation step* into another configuration $C_2 : V \to \{1, \ldots, N, \perp\}$, denoted $C_1 \to C_2$, if $C_1^{-1}(\ell) = C_2^{-1}(\ell)$ or $(C_1^{-1}(\ell), C_2^{-1}(\ell)) \in E$ holds for all $\ell \in \{1, \ldots, N\}$, i.e., if each robot does not move or moves to one of the at most four adjacent squares. Furthermore, two robots cannot exchange their squares in one transformation step, i.e., for all occupied squares $v \neq w \in V$, we require that $C_2(v) = C_1(w)$ implies $C_2(w) \neq C_1(v)$. For $M \in \mathbb{N}$, a *schedule* is a sequence $C_1 \to \cdots \to C_M$ of transformations. The number of steps in a schedule is called its *makespan*. Given a start configuration C_s and a target configuration C_t, the *optimal makespan* is the minimum number of steps in a schedule starting with C_s and ending with C_t. Let $n > 1$. Note that for the 2×2-, $1 \times n$- and $n \times 1$-rectangles, there are pairs of start and target configurations where no such sequence exists. For all other rectangles, such configurations do not exist; we provide an $\mathcal{O}(1)$-approximation of the makespan in Section 3.

For the continuous setting of Section 5, we consider N robots $R := \{1, \ldots, N\} \subseteq \mathbb{N}$. The Euclidean distance between two points $p, q \in \mathbb{R}^2$ is $|pq| := ||p - q||_2$. Every robot r has a *start* and *target position* $s_r, t_r \in \mathbb{R}^2$ with $|s_i s_j|, |t_i t_j| \geq 2$ for all $i \neq j$. In the following, $d := \max_{r \in R} |s_r t_r|$ is the maximum distance a robot has to cover. A *trajectory* of a robot r is a curve $m_r : [0, T_r] \to \mathbb{R}^2$, where $T_r \in \mathbb{R}^+$ denotes the *travel time* of r. This curve m_r does not have to be totally differentiable, but must be totally left- and right-differentiable. Intuitively, at any point in time, a robot has a unique *past* and *future direction* that are not necessarily identical. This allows the robot to make sharp turns, but does not allow jumps. We bound the speed of the robot by 1, i.e., for each point in time, both left and right derivative of m_r have Euclidean length at most 1. Let $m_i : [0, T_i] \to \mathbb{R}^2$ and $m_j : [0, T_j] \to \mathbb{R}^2$ be two trajectories; w.l.o.g., all travel times are equal to the maximum travel time $T_{\max}$ by extending m_r with $m_r(t) = m_r(T_r)$ for all $T_r < t \leq T_{\max}$. The trajectories m_i and m_j are *compatible* if the corresponding robots do not intersect at any time, i.e., if $|m_i(t)m_j(t)| \geq 2$ holds for all $t \in [0, T_i]$. A *trajectory set* of R is a set of compatible trajectories $\{m_1, \ldots, m_N\}$, one for each robot. The *(continuous) makespan* of a trajectory set $\{m_1, \ldots, m_N\}$ is defined as $\max_{r \in R} T_r$. A trajectory set $\{m_1, \ldots, m_N\}$ *realizes* a pair of start and target configurations $\mathcal{S} := (\{s_1, \ldots, s_N\}, \{t_1, \ldots, t_N\})$ if $m_r(0) = s_r$ and $m_r(T_r) = t_r$ hold for all $r \in R$. We are searching for a trajectory set $\{m_1, \ldots, m_N\}$ realizing $\mathcal{S}$ with minimal makespan.

3 Labeled grid permutation

Let $n_1 \geq n_2 \geq 2$, $n_1 \geq 3$ and let P be an $n_1 \times n_2$-rectangle. In this section, we show that computing the optimal makespan of arbitrarily chosen start and target configurations C_s and C_t of k robots in P is strongly NP-complete. This is followed by a $\mathcal{O}(1)$-approximation for the makespan.

▶ **Theorem 1.** *The minimum makespan parallel motion planning problem on a grid is strongly NP-hard.*

We prove hardness using a reduction from Monotone 3-Sat. Intuitively speaking, given a formula, we construct a parallel motion planning instance with a *variable robot* for each variable in the formula. To encode a truth assignment, each variable robot is forced to move

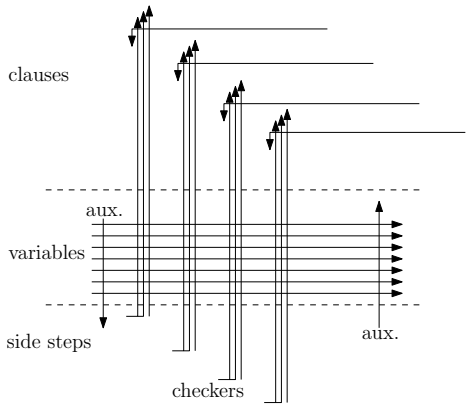


Figure 1 A sketch of the parallel motion planning instance resulting from the reduction.

on one of two paths. This is done by employing two groups of auxiliary robots that have to move towards their goal in a straight line in order to realize the given makespan. These auxiliary robots form *moving obstacles* whose position is known at any point in time.

The variable robots cross paths with *checker robots*, one for each literal of the formula, forcing the checker to wait for one time step if the assignment does not satisfy the literal. The checker robots then cross paths with *clause robots*; each clause robot has to move to its goal without delay and can only do so if at least one of the checkers did not wait. In order to ensure that the checkers meet with the clauses at the right time, further auxiliary robots force the checkers to perform a sequence of side steps in the beginning. Figure 1 gives a rough overview of the construction; full details of the proof are given in the full version of the paper [6].

In the proof of NP-completeness, we use a pair of start and target configurations in which the corresponding grids are not fully occupied. However, for our constant-factor approximation, we assume in Theorem 3 that the grid is fully occupied. This assumption is without loss of generality; our approximation algorithm works for any grid population, see Theorem 4.

Our constant-factor approximation is based on an algorithm that computes a schedule with a makespan upper-bounded by $\mathcal{O}(n_1 + n_2)$ described by Lemma 2. Based on Lemma 2, we give a constant factor approximation of the makespan, see Theorem 3. Finally, we embed the algorithm of Theorem 3 into a more general approach to ensure simultaneously a polynomial running time w.r.t. the number N of input robots and a constant approximation factor, see Theorem 4.

▶ **Lemma 2.** *For a pair of start and target configurations C_s and C_t of an $n_1 \times n_2$-rectangle, we can compute in polynomial time w.r.t. n_1 and n_2 a sequence of $\mathcal{O}(n_1 + n_2)$ steps transforming C_s into C_t.*

The high-level idea of the algorithm of Lemma 2 is the following. We apply a sorting algorithm called RotateSort [15] that computes a corresponding permutation of an $n_1 \times n_2$ (orthogonal) grid within $\mathcal{O}(n_1 + n_2)$ *parallel steps*. Each parallel step is made up of a set of pairwise disjoint *swaps*, each of which causes two neighbouring robots to exchange their positions. Because in our model direct swaps are not allowed, we simulate one parallel step by a sequence of $\mathcal{O}(1)$ transformation steps. This still results in a sequence of $\mathcal{O}(n_1 + n_2)$ transformation steps. A detailed description of the algorithm used in the proof of Lemma 2

is given in the full version of the paper [6]. An alternative to our Lemma 2 was recently proposed by Yu [27], who uses a routine called SplitAndGroup for achieving a makespan that is linear in the diameter of the rectangular environment.

Based on the algorithm of Lemma 2, we can give a constant-factor approximation algorithm.

▶ **Theorem 3.** *There is an algorithm with running time $\mathcal{O}(dn_1n_2)$ that, given an arbitrary pair of start and target configurations of an $n_1 \times n_2$-rectangle with maximum distance d between any start and target position, computes a schedule of makespan $\mathcal{O}(d)$, i.e., an approximation algorithm with constant stretch.*

For the algorithm of Theorem 3, Lemma 2 is repeatedly applied to rectangles of side length $\mathcal{O}(d)$, resulting in $\mathcal{O}(d)$ transformation steps in total. Because d is a lower bound on the makespan, this yields an $\mathcal{O}(1)$-approximation of the makespan.

At a high level, the algorithm of Theorem 3 first computes the maximal Manhattan distance d between a robot's start and target position. Then we partition P into a set T of pairwise disjoint rectangular *tiles*, where each tile $t \in T$ is an $n'_1 \times n'_2$-rectangle for $n'_1, n'_2 \leq 24d$. We then use an algorithm based on flows to compute a sequence of $\mathcal{O}(d)$ transformation steps, ensuring that all robots are in their target tile. Once all robots are in the correct tile, we use Lemma 2 simultaneously on all tiles to move each robot to the correct position within its target tile. The details of the algorithm of Theorem 3 are given further down in this section.

The above mentioned tiling construction ensures that each square of P belongs to one unambiguously defined tile and each robot has a *start* and *target tile*.

Based on the approach of Theorem 3 we give a $\mathcal{O}(1)$-approximation algorithm for the makespan with a running time polynomial w.r.t. the number N of robots to be moved.

▶ **Theorem 4.** *There is an algorithm with running time $\mathcal{O}(N^5)$ that, given an arbitrary pair of start and target configurations of a rectangle P with N robots to be moved and maximum distance d between any start and target position, computes a schedule of makespan $\mathcal{O}(d)$, i.e., an approximation algorithm with constant stretch.*

Intuitively speaking, the approach of Theorem 4 distinguishes two cases.

(1) Both $\lfloor \frac{n_2}{4} \rfloor$ and the maximum distance d between the robots' start and target positions, are lower-bounded by the number N of input robots.

(2) $N > \lfloor \frac{n_2}{4} \rfloor$ or $N > d$.

In case (1), the grid is populated sparsely enough such that the robots' trajectories in northern, eastern, southern, and western direction can be done sequentially by four individual transformation sequences.

In order to ensure that each robot has locally enough space, we consider a preprocessed start configuration C_o in which the robots have odd coordinates. We ensure that C_s can be transformed into C_o within $\mathcal{O}(d)$ steps. Analogously, we ensure that the outcome of the northern, eastern, southern, and western trajectories is a configuration C_e with even coordinates, such that C_e can be transformed into C_t within $\mathcal{O}(d)$ transformation steps. The choice of the divisor 4 in the criteria "$N \leq \lceil \frac{n_1}{4} \rceil$" has the following technical reasoning: In the first case of the proof of Theorem 4, we assume w.l.o.g. that n_1 and n_2 are even. If this is not the case, we move all robots from the last line into the second-to-last line and from the last column into the second-to-last column. This may double the largest x-coordinate of a robot and the largest y-coordinate of a robot, e.g., in the case that all positions in the last line and all positions in that last column are occupied by robots. In a next step,

we transform the start configuration into a configuration C_o in which all robots' x- and y-coordinates are odd. This may cause another doubling of the largest x-coordinate and the largest y-coordinate which may result fourfold increase of the largest x-coordinate and a fourfold increase of the largest y-coordinate. The assumption $N \leq \lceil \frac{n_2}{4} \rceil$ ensures that both dimensions of the rectangular environment are large enough because $n_1 \geq n_2$.

In the second case, we apply the approach of Theorem 3 as a subroutine to a union of smallest rectangles that contain the robots' start and target configurations.

The full detailed version of the proof of Theorem 4 can be found in the full version of the paper [6].

In the rest of Section 3, we give the proof of Theorem 3, i.e. we give an algorithm that computes a schedule with makespan linear in the maximum distance between robots' start and target positions. The remainder of the proof of Theorem 3 is structured as follows. In Section 3.1 we give an outline of our *flow algorithm* that ensures that each robot reaches its target tile in $\mathcal{O}(d)$ transformation steps. Section 3.2 gives the full intuition of this algorithm and its subroutines. (For full details, we refer to the full version of the paper [6]).

3.1 Outline of the approximation algorithm of Theorem 3

We model the trajectories of robots between tiles as a flow f_T, using the weighted directed graph $G_T = (T, E_T, f_T)$, which is dual to the tiling T defined in the previous section. In G_T, we have an edge $(v, w) \in E_T$ if there is at least one robot that has to move from v into w. Furthermore, we define the weight $f_T((v, w))$ of an edge as the integer number of robots that move from v to w. As P is fully occupied, f_T is a *circulation*, i.e., a flow with no sources or sinks, in which flow conservation has to hold at all vertices. Because the side lengths of the tiles are greater than d, G_T is a grid graph with additional diagonal edges and thus has degree at most 8.

The maximum edge value of f_T is $\Theta(d^2)$, but only $\mathcal{O}(d)$ robots can possibly leave a tile within a single transformation step. Therefore, we decompose the flow f_T of robots into a *partition* consisting of $\mathcal{O}(d)$ *subflows*, where each individual robot's motion is modeled by exactly one subflow and each edge in the subflow has value at most d. Thus we are able to *realize* each subflow in a single transformation step by placing the corresponding robots adjacent to the boundaries of its corresponding tiles before we realize the subflow. To facilitate the decomposition into subflows, we first preprocess G_T. In total, the algorithm consists of the following subroutines, elaborated in detail in Section 3.2.

- **Step 1:** Compute d, the tiling T and the corresponding flow G_T.
- **Step 2:** Preprocess G_T in order to remove intersecting and bidirectional edges.
- **Step 3:** Compute a partition into $\mathcal{O}(d)$ d-subflows.
- **Step 4:** Realize the $\mathcal{O}(d)$ subflows using $\mathcal{O}(d)$ transformation steps.
- **Step 5:** Simultaneously apply Lemma 2 to all tiles, moving each robot to its target position.

3.2 Details of the approximation algorithm of Theorem 3

In this section we only give more detailed descriptions of Steps 1-4 because Step 5 is a trivial application of Lemma 2 to all tiles in parallel.

3.2.1 Step 1: Compute d, the tiling T, and the corresponding flow G_T

The maximal distance between robots' start and target positions can be computed in a straightforward manner.

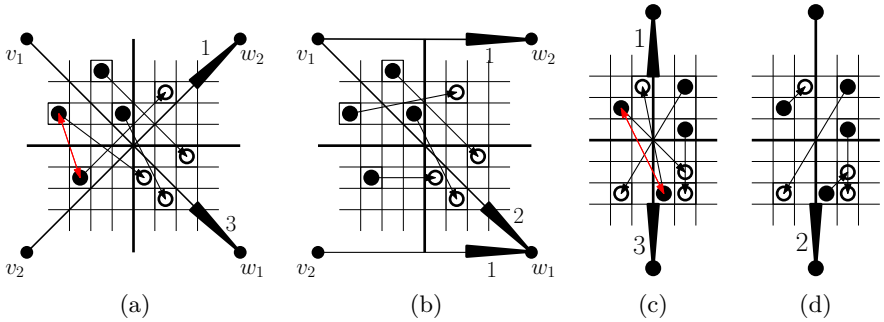


Figure 2 Illustration of the preprocessing (step (1): before and after removing crossing edges (a)+(b) and step (2): before and after removing bidirectional edges (c)+(d)). The red arrows indicate how robots change their positions during the preprocessing steps.

For the tiling, we assume that the rectangle P is axis aligned and that its bottom-left corner is $(0, 0)$. We consider $k_v := \lfloor \frac{n_1}{12d} \rfloor$ vertical lines $\ell_1^v, \ldots, \ell_{k_v}^v$ with x-coordinate modulo $12d$ equal to 0. Analogously, we consider $k_h := \lfloor \frac{n_2}{12d} \rfloor$ horizontal lines $\ell_1^h, \ldots, \ell_{k_h}^h$ with y-coordinate modulo $12d$ to 0. Finally, we consider the tiling of P that is induced by the arrangement induced by $\ell_1^v, \ldots, \ell_{k_v-1}^v, \ell_1^h, \ldots, \ell_{h_v-1}^h$ and the boundary of P. This implies that the side length of a tile is upper-bounded by $24d - 1$.

Finally, computing the flow G_T is straightforward by considering the tiling T and the robots' start and target positions.

3.2.2 Step 2: Ensuring planarity and unidirectionality

After initialization, we preprocess G_T, removing edge intersections and bidirectional edges by transforming the start configuration C_s into an intermediate start configuration C_s', obtaining a planar flow without bidirectional edges. This transformation consists of two steps: (1) ensuring planarity and (2) ensuring unidirectionality.

Step (1): We observe that edge crossings only occur between two diagonal edges with adjacent source tiles, as illustrated in Figure 2(a)+(b). To remove a crossing, it suffices to eliminate one of the diagonal edges by exchange robots between the source tiles. To eliminate all crossings, each robot is moved at most once, because after moving, the robot does no longer participate in a diagonal edge. Thus, all necessary exchanges can be done in $\mathcal{O}(d)$ steps by Lemma 2, covering the tiling T by constantly many layers, similar to the proof of Lemma 2.

Step (2): We delete a bidirectional edge $(v, w), (w, v)$ by moving $\min\{f_T((v, w)), f_T((w, v))\}$ robots with target tile w from v to w and vice versa which achieves that $\min\{f_T((v, w)), f_T((w, v))\}$ robots achieve their target tile w and $\min\{f_T((v, w)), f_T((w, v))\}$ robots achieve their target tile v, thus eliminating the edge with lower flow value. This process is depicted in Figure 2(c)+(d). Like step (1), this can be done in $\mathcal{O}(d)$ parallel steps by Lemma 2. As we do not add any edges, we maintain planarity during step (2). Observe that during the preprocessing, we do not destroy the grid structure of G_T.

Step (1) and step (2) maintain the flow property of f_T without any other manipulations to the flow f_T, because both preprocessing steps can be represented by local circulations.

3.2.3 Step 3: Computing a flow partition

After preprocessing, we partition the flow G_T into d-subflows.

▶ **Definition 5.** A *subflow* of G_T is a circulation $G'_T = (T, E', f'_T)$, such that $E' \subseteq E_T$, and $0 \leq f'_T(e) \leq f_T(e)$ for all $e \in E'$. If $f'_T(e) \leq z$ for all $e \in E'$ and some $z \in \mathbb{N}$, we call G'_T a z-flow.

The flow partition relies on an upper bound on the maximal edge weight in G_T. By construction, tiles have side length at most $24d$; therefore, each tile consists of at most $576d^2$ unit squares. This yields the following upper bound; a tighter constant factor can be achieved using a more sophisticated argument.

▶ **Observation 6.** *We have $f_T(e) \leq 576d^2$ for all $e \in E_T$.*

▶ **Definition 7.** A (z, ℓ)-partition of G_T is a set of ℓ z-subflows $\{G_1 = (V_1, E_1, f_1), \ldots, G_\ell = (V_\ell, E_\ell, f_\ell)\}$ of G_T, such that $G_1, \ldots, G_\ell$ sum up to G_T.

▶ **Lemma 8.** *We can compute a $(d, \mathcal{O}(d))$-partition of G_T in polynomial time.*

Proof sketch. In a slight abuse of notation, throughout this proof, the elements in sets of cycles are not necessarily unique. A $(d, \mathcal{O}(d))$-partition can be constructed using the following steps.

- We start by computing a $(1, h)$-partition $\mathbb{C}_{\bigcirc}$ of G_T consisting of $h \leq n_1 n_2$ cycles. This is possible because G_T is a circulation. If a cycle C intersects itself, we subdivide C into smaller cycles that are intersection-free. Furthermore, h is clearly upper bounded by the number of robots $n_1 n_2$, because every robot can contribute only 1 to the sum of all edges in G_T. As the cycles do not self-intersect, we can partition the cycles $\mathbb{C}_{\bigcirc}$ by their orientation, obtaining the set $\mathbb{C}_{\circlearrowright}$ of clockwise and the set $\mathbb{C}_{\circlearrowleft}$ of counterclockwise cycles.
- We use $\mathbb{C}_{\circlearrowright}$ and $\mathbb{C}_{\circlearrowleft}$ to compute a $(1, h')$-partition $\mathbb{C}^1_{\circlearrowright} \cup \mathbb{C}^2_{\circlearrowright} \cup \mathbb{C}^1_{\circlearrowleft} \cup \mathbb{C}^2_{\circlearrowleft}$ with $h' \leq n_1 n_2$, such that two cycles from the same subset $\mathbb{C}^1_{\circlearrowright}$, $\mathbb{C}^2_{\circlearrowright}$, $\mathbb{C}^1_{\circlearrowleft}$, or $\mathbb{C}^2_{\circlearrowleft}$ share a common orientation. Furthermore, we guarantee that two cycles from the same subset are either edge-disjoint or one lies nested in the other. A partition such as this can be constructed by applying a recursive peeling algorithm to $\mathbb{C}_{\circlearrowright}$ and $\mathbb{C}_{\circlearrowleft}$ as depicted in Figure 3, yielding a decomposition of the flow induced by $\mathbb{C}_{\circlearrowright}$ into two cycle sets $\mathbb{C}^1_{\circlearrowright}$ and $\mathbb{C}^2_{\circlearrowright}$, where $\mathbb{C}^1_{\circlearrowright}$ consists of clockwise cycles and $\mathbb{C}^2_{\circlearrowright}$ consists of counterclockwise cycles, and a similar partition of $\mathbb{C}_{\circlearrowleft}$, see the appendix for details.
- Afterwards, we partition each set $\mathbb{C}^1_{\circlearrowright}$, $\mathbb{C}^2_{\circlearrowright}$, $\mathbb{C}^1_{\circlearrowleft}$, and $\mathbb{C}^2_{\circlearrowleft}$ into $\mathcal{O}(d)$ subsets, each inducing a d-subflow of G_T, see the appendix for details. ◀

3.2.4 A subroutine of Step 4: Realizing a single subflow

In this section, we present a procedure for *realizing* a single d-subflow G'_T of G_T.

▶ **Definition 9.** A schedule $t := C_1 \to \cdots \to C_{k+1}$ *realizes* a subflow $G'_T = (T, E', f'_T)$ if, for each pair v, w of tiles, the number of robots moved by t from their start tile v to their target tile w is $f'_T((v, w))$, where we let $f'_T((v, w)) = 0$ if $(v, w) \notin E'$.

▶ **Lemma 10.** *Let $G'_T = (T, E'_T, f'_T)$ be a planar unidirectional d-subflow. There is a polynomial-time algorithm that computes a schedule $C_1 \to \cdots \to C_{k+1}$ realizing G'_T for a constant $k \in \mathcal{O}(1)$.*

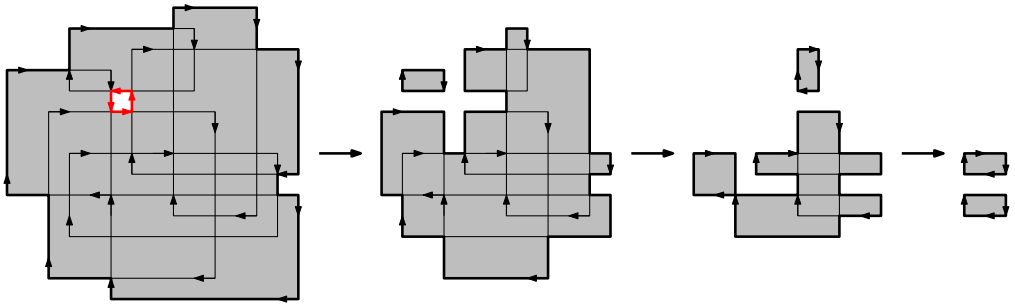
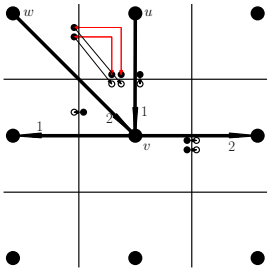
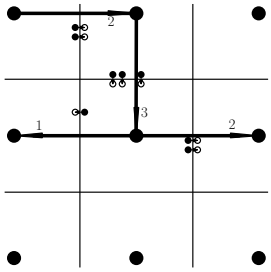


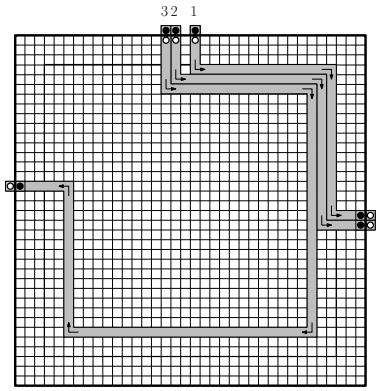
Figure 3 Recursive peeling of the area bounded by the cycles from $\mathbb{C}_{\circlearrowleft}$, resulting in clockwise cycles (thick black cycles). Cycles constituting the boundary of *holes* are counterclockwise (thick red cycles). Note that an edge e vanishes when $f_T(e)$ cycles containing that edge are removed by the peeling algorithm described above.



(a) Preprocessing of diagonal edges.



(b) Configuration and flow after preprocessing.



(c) A crossing-free matching of incoming and outgoing robots and the connecting paths inside the corresponding tile, for $d = 3$.

Figure 4 Procedure for computing transformation steps that realize a d-subflow. Figures (a) and (b) illustrate how we preprocess G'_T such that E'_T consists of horizontal and vertical edges only. Figure (c) illustrates the main approach. White disks illustrate start positions and black disks illustrate target positions.

Proof sketch. We give a high-level description of the proof and refer to the full version of the paper [6] for details.

Our algorithm uses $k = \mathcal{O}(d)$ preprocessing steps $C_1 \to \cdots \to C_k$, as depicted in Figure 4(a)+(b), and one final realization step $C_k \to C_{k+1}$, shown in Figure 4(c), pushing the robots from their start tiles into their target tiles. The preprocessing eliminates diagonal edges and places the moving robots next to the border of their target tiles. For the final realization step we compute a pairwise disjoint matching between incoming and outgoing robots, such that each pair is connected by a tunnel inside the corresponding tile in which these tunnels do not intersect, see Figure 4(a). The final realization step is given via the robots' motion induced by pushing each robot into the interior of the tile and by pushing this one-step motion through the corresponding tunnel into the direction of the corresponding outgoing robot. ◀

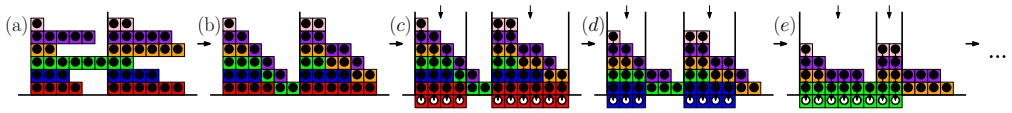


Figure 5 Stacking robots in lines induced by flows of the edges of the subflows to be realized.

3.2.5 Step 4: Realizing all subflows

Next we extend the idea of Lemma 10 to $\ell \leq d$ subflows instead of one and demonstrate how this can be leveraged to move all robots to their target tile using $\mathcal{O}(d)$ transformation steps.

▶ **Lemma 11.** *Let $\mathcal{S} := \langle G_1 = (V_1, E_1, f_1), \ldots, G_\ell = (V_\ell, E_\ell, f_\ell)\rangle$ be a sequence of $\ell \leq d$ unidirectional planar d-subflows of G_T. There is a polynomial-time algorithm computing $\mathcal{O}(d) + \ell$ transformation steps $C_1 \to \cdots \to C_{k+\ell}$ realizing $\mathcal{S}$.*

Proof sketch. We give a high level description of the proof and refer for details to the full version of the paper [6].

Let t be an arbitrary tile. Similar to the approach of Lemma 10, we first apply a preprocessing step guaranteeing that the robots to be moved into or out of t are in the right position close to the boundary of t, see Figure 5. Thereafter we move the robots into their target tiles, using ℓ applications of the algorithm from Lemma 10 without the preprocessing phase. In particular, we realize a sequence of ℓ d-subflows by applying ℓ times the single realization step of Lemma 10. ◀

▶ **Lemma 12.** *There is a polynomial-time algorithm computing $\mathcal{O}(d)$ transformation steps moving all robots into their target tiles.*

Proof. By Lemma 8, we can compute a (d, cd)-partition of G_T for $c \in \mathcal{O}(1)$. We group the corresponding d-subflows into $\frac{cd}{d} = c$ sequences, each consisting of at most d d-subflows. We realize each sequence by applying Lemma 11, using $\mathcal{O}(d)$ transformation steps for each sequence. This leads to $\mathcal{O}(cd) = \mathcal{O}(d)$ steps for realizing all sequences of d-subflows. ◀

For the proof of Theorem 3, we still need to analyze the time complexity of our approach, for which we refer to the full version of the paper [6].

4 Variants on labeling

A different version is the unlabeled variant, in which all robots are the same. A generalization of both this and the labeled version arises when robots belong to one of k color classes, with robots from the same color class being identical.

We formalize this problem variant by using a coloring $c : \{1, \ldots, n_1 n_2\} \to \{1, \ldots, k\}$ for grouping the robots. By populating unoccupied cells with robots carrying color $k + 1$, we may assume that each unit square in the environment P is occupied. The robots *draw an image* $I = \left(I^1, \ldots, I^k\right)$, where I^i is the set of cells occupied by a robot with color i. We say that two images I_s and I_t are *compatible* if in I_s and I_t the number of cells colored with color i are equal for each color $i = 1, \ldots, k$. By moving the robots, we want to transform a start image I_s into a compatible target image I_t, minimizing the makespan.

▶ **Theorem 13.** *There is an algorithm with running time $\mathcal{O}(k(N)^{1.5} \log(N) + N^5)$ for computing, given start and target images I_s, I_t with maximum distance d between start and target positions, an $\mathcal{O}(1)$-approximation of the optimal makespan M and a corresponding schedule.*

The basic idea is to transform the given unlabeled problem setting into a labeled problem setting by solving a geometric bottleneck matching problem, see the the full version of the paper [6] for details.

5 Continuous motion

The continuous case considers N unit disks that have to move into a target configuration; the velocity of each robot is bounded by 1, and we want to minimize the makespan. For arrangements of disks that are not well separated, we show that constant stretch is *impossible*.

▶ **Theorem 14.** *There is an instance with optimal makespan $M \in \Omega(N^{1/4})$.*

The basic proof idea is as follows. Let $\{m_1, \ldots, m_N\}$ be an arbitrary trajectory set with makespan M. We show that there must be a point in time $t \in [0, M]$ where the area of $\mathrm{Conv}(m_1(t), \ldots, m_N(t))$ is lower-bounded by $cN + \Omega(N^{3/4})$, where cN is the area of the convex hull $\mathrm{Conv}(m_1(0), \ldots, m_N(0))$ of $m_1(0), \ldots, m_N(0)$. Assume $M \in o\left(N^{1/4}\right)$ and consider the area of $\mathrm{Conv}(m_1(t'), \ldots, m_N(t'))$ at some point $t' \in [0, M]$. This area is at most $cN + \mathcal{O}(\sqrt{N}) \cdot o\left(N^{1/4}\right)$ which is a contradiction. Proof details are given in the full version of the paper [6].

Conversely, we give a non-trivial upper bound on the stretch, as follows.

▶ **Theorem 15.** *There is an algorithm that computes a trajectory set with continuous makespan of $\mathcal{O}(d + \sqrt{N})$. If $d \in \Omega(1)$, this implies a $\mathcal{O}(\sqrt{N})$-approximation algorithm.*

The approach of Theorem 15 applies an underlying grid with mesh size $2\sqrt{2}$. Our algorithm (1) moves the robots to vertices of the grid, (2) applies our $\mathcal{O}(1)$-approximation for the discrete case, and (3) moves the robots from the vertices of the grid to their targets. For a detailed description of the Algorithm of Theorem 15 see the full version of the paper [6].

6 Conclusion

We have presented progress on a number of algorithmic problems of parallel motion planning, also shedding light on a wide range of interesting open problems described in the following.

The first set of problems consider complexity. The labeled problem of Section 3 is known to be NP-complete for planar graphs. It is natural to conjecture that the geometric version is also hard. It seems tougher to characterize the family of optimal trajectories: As shown above, their nature is unclear, so membership in NP is doubtful.

A second set of questions considers the relationship between stretch factor and disk separability in the continuous setting. We believe that the upper bound of $\mathcal{O}(\sqrt{N})$ on the worst-case stretch factor for dense arrangements is tight. What is the critical separability of disks for which constant stretch can be achieved? How does the stretch factor increase as a function of N below this threshold? For *sparse* arrangements of disks, simple greedy, straight-line trajectories between the origins and destinations of disks encounter only isolated conflicts, resulting in small stretch factors close to 1, i.e., $1 + o(1)$. What is the relationship between (local) density and the achievable stretch factor along the whole density spectrum?

Finally, practical motion planning requires a better handle on characterizing and computing optimal solutions for specific instances, along with lower bounds, possibly based on numerical methods and tools. Moreover, there is a wide range of additional objectives and requirements, such as accounting for acceleration or deceleration of disks, turn cost, or multi-stop tour planning. All these are left for future work.

References

1. M. Abellanas, S. Bereg, F. Hurtado, A. G. Olaverri, D. Rappaport, and J. Tejel. Moving coins. *Computational Geometry: Theory and Applications*, 34(1):35–48, 2006.
2. Aviv Adler, Mark de Berg, Dan Halperin, and Kiril Solovey. Efficient multi-robot motion planning for unlabeled discs in simple polygons. *IEEE Transactions on Automation Science and Engineering*, 12(4):1309–1317, 2015.
3. B. Aronov, M. de Berg, A. F. van der Stappen, P. Švestka, and J. Vleugels. Motion planning for multiple robots. *Discrete & Computational Geometry*, 22(4):505–525, 1999.
4. Aaron T. Becker, Sándor P. Fekete, Phillip Keldenich, Lillian Lin, and Christian Scheffer. Coordinated motion planning: The video. In Csaba Tóth and Bettina Speckmann, editors, *34th International Symposium on Computational Geometry (SoCG 2018, these proceedings)*, volume 99 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 74:1–74:6, 2018. Video available via https://youtu.be/0OrG72sX5gk.
5. S. Bereg, A. Dumitrescu, and J. Pach. Sliding disks in the plane. *International Journal of Computational Geometry & Applications*, 18(5):373–387, 2008.
6. Erik D. Demaine, Sándor P. Fekete, Phillip Keldenich, Henk Meijer, and Christian Scheffer. Coordinated motion planning: Reconfiguring a swarm of labeled robots with bounded stretch. *Computing Research Repository (CoRR)*, 1801:1–32, 2018. Available at https://arxiv.org/abs/1801.01689.
7. A. Dumitrescu and M. Jiang. On reconfiguration of disks in the plane and related problems. *Computational Geometry: Theory and Applications*, 46:191–202, 2013.
8. G. W. Flake and E. B. Baum. Rush Hour is PSPACE-complete, or "Why you should generously tip parking lot attendants". *Theoretical Computer Science*, 270(1):895–911, 2002.
9. R. A. Hearn and E. D. Demaine. PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science*, 343(1):72–96, 2005.
10. R. A. Hearn and E. D. Demaine. *Games, puzzles, and computation.* CRC Press, 2009.
11. J. E. Hopcroft, J. T. Schwartz, and M. Sharir. On the complexity of motion planning for multiple independent objects; PSPACE-hardness of the warehouseman's problem. *International Journal of Robotics Research*, 3(4):76–88, 1984.
12. J. E. Hopcroft and G. T. Wilfong. Reducing multiple object motion planning to graph searching. *SIAM Journal on Computing*, 15(3):768–785, 1986.
13. S. Kloder and S. Hutchinson. Path planning for permutation-invariant multi-robot formations. In *IEEE Transactions on Robotics*, volume 22, pages 650–665. IEEE, 2006.
14. D. Kornhauser, G. Miller, and P. Spirakis. Coordinating pebble motion on graphs, the diameter of permutation groups, and applications. In *Proceedings of the 25th Annual Symposium on Foundations of Computer Science (SFCS)*, pages 241–250, 1984. doi:10.1109/SFCS.1984.715921.
15. J. M. Marberg and E. Gafni. Sorting in constant number of row and column phases on a mesh. *Algorithmica*, 3:561–572, 1988. doi:10.1007/BF01762132.
16. Mark Overmars. Contributed open problem. In Sándor P. Fekete, Rudolf Fleischer, Rolf Klein, and Alejandro Lopez-Ortiz, editors, *Robot Navigation, Dagstuhl Seminar 06421*, 2006. URL: http://www.dagstuhl.de/de/programm/kalender/semhp/?semnr=06421.
17. G. Ramanathan and V. Alagar. Algorithmic motion planning in robotics: Coordinated motion of several disks amidst polygonal obstacles. In *Proceedings of the Second IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 514–522, 1985.
18. D. Ratner and M. K. Warmuth. Finding a shortest solution for the $N \times N$ extension of the 15-puzzle is intractable. In *Proceedings of the Fifth AAAI Conference on Artificial Intelligence*, pages 168–172, 1986.

19 Christian Scheideler. *Universal routing strategies for interconnection networks*, volume 1390 of *Lecture Notes in Computer Science*. Springer, 1998.

20 Jacob T. Schwartz and M. Sharir. On the piano movers' problem: III. Coordinating the motion of several independent bodies: the special case of circular bodies moving amidst polygonal barriers. *International Journal of Robotics Research*, 2(3):46–75, 1983.

21 K. Solovey and D. Halperin. k-color multi-robot motion planning. *International Journal of Robotics Research*, 33(1):82–97, 2014.

22 K. Solovey, J. Yu, O. Zamir, and D. Halperin. Motion planning for unlabeled discs with optimality guarantees. In *Robotics: Science and Systems (RSS)*, 2015.

23 Kiril Solovey and Dan Halperin. On the hardness of unlabeled multi-robot motion planning. *International Journal of Robotics Research*, 35(14):1750–1759, 2016.

24 P. Spirakis and C. K. Yap. Strong NP-hardness of moving many discs. *Information Processing Letters*, 19(1):55–59, 1984.

25 M. Turpin, N. Michael, and V. Kumar. Trajectory planning and assignment in multirobot systems. In *Algorithmic Foundations of Robotics X*, pages 175–190. Springer, 2013.

26 R. M. Wilson. Graph puzzles, homotopy, and the alternating group. *Journal of Combinatorial Theory, Series B*, 16(1):86–96, 1974.

27 J. Yu. Constant factor optimal multi-robot path planning in well-connected environments. arXiv. URL: `https://arxiv.org/abs/1706.07255`.

28 J. Yu. Constant factor time optimal multi-robot routing on high-dimensional grids in mostly sub-quadratic time. arXiv. URL: `https://arxiv.org/abs/1801.10465`.

3D Snap Rounding

Olivier Devillers
Université de Lorraine, CNRS, Inria, LORIA
F-54000 Nancy, France

Sylvain Lazard
Université de Lorraine, CNRS, Inria, LORIA
F-54000 Nancy, France
sylvain.lazard@inria.fr

William J. Lenhart
Williams College
Williamstown, Massachusetts, USA

Abstract

Let $\mathcal{P}$ be a set of n polygons in $\mathbb{R}^3$, each of constant complexity and with pairwise disjoint interiors. We propose a rounding algorithm that maps $\mathcal{P}$ to a simplicial complex $\mathcal{Q}$ whose vertices have integer coordinates. Every face of $\mathcal{P}$ is mapped to a set of faces (or edges or vertices) of $\mathcal{Q}$ and the mapping from $\mathcal{P}$ to $\mathcal{Q}$ can be done through a continuous motion of the faces such that (i) the L_∞ Hausdorff distance between a face and its image during the motion is at most $3/2$ and (ii) if two points become equal during the motion, they remain equal through the rest of the motion. In the worst case, the size of $\mathcal{Q}$ is $O(n^{15})$ and the time complexity of the algorithm is $O(n^{19})$ but, under reasonable hypotheses, these complexities decrease to $O(n^5)$ and $O(n^6\sqrt{n})$.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Geometric algorithms, Robustness, Fixed-precision computations

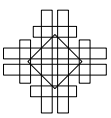
Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.30

Related Version A full version of this paper is available at `https://hal.inria.fr/hal-01698928`

Acknowledgements The authors would like to thank A. Lieutier, H. Everett, M. de Berg, D. Halperin, R. Seidel, and D. Bremner for many discussions on the problems.

1 Introduction

Rounding 3D polygonal structures is a fundamental problem in computational geometry. Indeed, many implementations dealing with 3D polygonal objects, in academia and industry, require as input pairwise-disjoint polygons whose vertices have coordinates given with fixed-precision representations (usually with 32 or 64 bits). On the other hand, many algorithms and implementations dealing with 3D polygonal objects in computational geometry output polygons whose vertices have coordinates that have arbitrary-precision representations. For instance, when computing boolean operations on polyhedra, some new vertices are defined as the intersection of three faces and their exact coordinates are rational numbers whose numerators and denominators are defined with roughly three times the number of bits used for representing each input coordinate. When applying a rotation to a polyhedron, the new vertices have coordinates that involve trigonometric functions. When sampling algebraic surfaces, the vertices are obtained as solutions of algebraic systems and they may require arbitrary-precision representations since the distance between two solutions may be arbitrarily small (depending on the degree of the surface).




This discrepancy between the precision of the input and output of many geometric algorithms is an issue, especially in industry, because it often prevents the output of one algorithm from being directly used as the input to a subsequent algorithm.

In this context, there exists no solution for rounding the coordinates of 3D polygons with the constraint that their rounded images do not properly intersect and that every input polygon and its rounded image remain close to each other (in Hausdorff distance). In practice, coordinates are often rounded without guarding against changes in topology and there is no guarantee that the rounded faces do not properly intersect one another.

The same problem in 2D for segments, referred to as snap rounding, has been widely studied and admits practical and efficient solutions [1, 5–11, 14]. Given a set of possibly intersecting segments in 2D, the problem is to subdivide their arrangement and round the vertices so that no two disjoint segments map to segments that properly intersect. For clarity, all schemes consider that vertices are rounded on the integer grid. It is well known that rounding the endpoints of the edges of the arrangement to their closest integer point is not a good solution because it may map disjoint segments to properly intersecting segments.. Snap rounding schemes propose to further split the edges when they share a pixel (a unit square centered on the integer grid). In such schemes, disjoint edges may collapse but this is inevitable if the rounding precision is fixed and if we bound the Hausdorff distance between the edges and their rounded images. Furthermore, it is NP-hard to determine whether it is possible to round simple polygons with fixed precision and bounded Hausdorff distance, and without changing the topological structure [12].

In dimension three, results are extremely scarse, despite the significance of the problem. Goodrich et al. [5] proposed a scheme for rounding segments in 3D, and Milenkovic [13] sketched a scheme for polyhedral subdivisions but, as pointed out by Fortune [4], both schemes have the property that rounded edges can cross. Fortune [3] suggested a high-level rounding scheme for polyhedra but in a specific setting that does not generalize to polyhedral subdivisions [4]. Finally, Fortune [4] proposed a rounding algorithm that maps a set $\mathcal{P}$ of n disjoint triangles in $\mathbb{R}^3$ to a set $\mathcal{Q}$ of triangles with $O(n^4)$ vertices on a discrete grid such that (i) every triangle of $\mathcal{P}$ is mapped to a set of triangles in $\mathcal{Q}$ at L_∞ Hausdorff distance at most $\frac{3}{2}$ from the original face and (ii) the mapping preserves or collapses the vertical ordering of the faces. Unfortunately, this rounding scheme is very intricate and, moreover, it uses a grid precision that depends on the number n of triangles: the vertices coordinates are rounded to integer multiples of about $\frac{1}{n}$.

The difficulty of snap rounding faces in 3D is described by Fortune [4]: First, it is reasonable to round every vertex to the center of the voxel containing it (a voxel is a unit cube centered on the integer grid). But, by doing so, a vertex may traverse a face and to avoid that, it might be necessary to add beforehand a vertex on the face, which requires triangulating it. Newly formed edges may cross older edges when snapping; to avoid this, new vertices are added to these edges, in turn requiring further triangulating of faces. It is not known whether such schemes terminate.

To better understand the difficulty of the problem, consider the following simple but flawed algorithm. First project all the input faces onto the horizontal plane, subdivide the projected edges as in 2D snap rounding, triangulate the resulting arrangement, lift this triangulation vertically on all faces, and then round all vertices to the centers of their voxels. For an input of size n, this yields an output of size $\Theta(n^4)$ in the worst case and an L_∞ Hausdorff distance of at most $\frac{1}{2}$ between the input faces and their rounded images. Unfortunately, this algorithm does not work in the sense that edges may cross: indeed, consider two almost vertical close triangles whose projections on the horizontal plane are

disjoint triangles that are rounded in 2D to the same segment; such triangles in 3D may be rounded into properly overlapping triangles. Fortune [4] solved this problem by using a finer grid to round the vertices and to avoid vertical rounding of the faces.

Contributions. We present in this paper the first algorithm for rounding a set of interior-disjoint polygons into a simplicial complex whose vertices have integer coordinates and such that the geometry does not change too much: namely, (i) the Hausdorff distance between every input face and its rounded image is bounded by a constant ($\frac{3}{2}$ for the L_∞ metric) and (ii) the relative positions of the faces are preserved in the sense that there is a continuous motion that deforms all input faces into their rounded images such that if two points collapse at some time, they remain identical up to the end of the motion (see Thm. 1). This ensures, in particular, that if a line stabs two input faces far enough from their boundaries, the line will stab their rounded images in the same order or in the same point.

The worst-case complexity of our algorithm is polynomial but unsatisfying as our upper bound on the output simplicial complex is $O(n^{15})$ for an input of size n (see Prop. 7). However, this upper bound decreases to $O(n^5)$ under the assumption that, roughly speaking, the input is a nice discretization of a constant number of surfaces that satisfy some reasonable hypothesis on their curvature (see Prop. 13 for details). The corresponding time complexity reduces from $O(n^{19})$ to $O(n^6\sqrt{n})$. It is also very likely that these bounds are not tight and, in practice on realistic non-pathological data, we anticipate time and space complexities of $O(n\sqrt{n})$ (see Remark 14).

Notation. The coordinates in Euclidean space $\mathbb{R}^3$ are referred to as x, y, and z and $\vec{\imath}, \vec{\jmath}, \vec{k}$ is the canonical basis. We use several planes parallel to the axes to project or intersect some faces: the xy-plane is called the *floor*, the xz-plane is called the *back wall* and a plane parallel to the yz-plane is called a *side wall*. Projections on the floor and on the back wall are always considered orthogonal to the plane of projection. Two polygons, edges, or vertices are said to *properly intersect* if their intersection is non-empty and not a common face of both.

General position assumption. For the sake of simplicity, we assume, without loss of generality, some general position on our input set of polygons $\mathcal{P}$. Precisely, we assume:

(α) No faces are parallel to the axes of coordinates and no vertices project along the y-axis on an edge (except the endpoints of that edge).

(β) No supporting plane of a face translated by vector $\vec{\jmath}$ or $-\vec{\jmath}$ contains a vertex.

Let $\mathcal{I}$ denote the intersection, if not empty, of the supporting plane of a face with the translation by $\pm\vec{\jmath}$ of the supporting plane of another face. By assumption (β), $\mathcal{I}$ is a line.

(γ) No vertices project along the y-axis onto such a line $\mathcal{I}$.

(δ) For any point A on a face and with half-integer x and y-coordinates, $A \pm \vec{\jmath}$ does not belong to another face. More generally, no line $\mathcal{I}$ crosses any vertical line defined by half-integer x and y-coordinates.

This general position assumption is done with no loss of generality because it can be achieved by a sequence of four symbolic perturbations of decreasing importance: (i) the input faces are translated in the x-direction by ϵ_1, (ii) translated in the y-direction by ϵ_2, (iii) the vector $\vec{\jmath}$ is scaled by a factor $(1+\epsilon_3)$, and (iv) the faces are rotated by an angle ϵ_4 around a line that is not parallel to the coordinate axes. As shown below, enforcing $\epsilon_1 \gg \epsilon_2 \gg \epsilon_3 \gg \epsilon_4$ yields that our perturbation scheme removes all degeneracies.

Consider an intersection $\mathcal{I}$ as defined above; $\mathcal{I}$ can be a line or a plane. If $\mathcal{I}$ is a line L that induces a degeneracy of type (δ), this degeneracy is avoided by a translation (i)

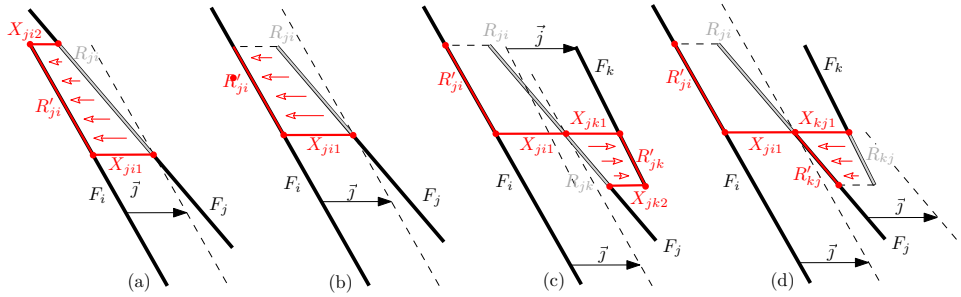


Figure 1 View inside a side wall $x = \text{cst}$. (a-d): The face F_j is partially projected onto F_i $(i < j)$, i.e., R_{ji} is replaced by the faces R'_{ji}, X_{ji1} and, in (a), X_{ji2}. (c): F_j is also partially projected onto F_k $(i < k < j)$. (d): If instead $i < j < k$, it is F_k that is partially projected onto F_j.

in the x-direction if L is not parallel to the xz-plane, and by a translation (ii) in the y-direction, otherwise. Then, perturbations (iii) and (iv) of smaller scales do not reintroduce this degeneracy [2]. If the intersection $\mathcal{I}$ is a plane, this remains the case after perturbations (i) and (ii), but the intersection becomes empty after a small enough perturbation (iii) and it remains empty after perturbation (iv). Hence, degeneracies of type (δ) are avoided by our scheme of perturbations.

Degeneracies of type (β) and (γ) are not affected by perturbations (i) and (ii), but they are avoided by the scaling of type (iii). Indeed, if $\mathcal{I}$ is a line then, viewed in projection on the back wall, the scaling of type (iii) translates the line. Finally, degeneracies of type (α) are not affected by perturbations (i-iii), but they are avoided by a rotation of type (iv).

2 Algorithm

We first describe the main algorithm in Section 2.1 and then two algorithmic refinements in Section 2.2 that we present separately for clarity. Our algorithm has the following property.

▶ **Theorem 1.** *Given a set $\mathcal{P}$ of polygonal faces in 3D in general position and that do not properly intersect, the algorithm outputs a simplicial complex $\mathcal{Q}$ whose vertices have integer coordinates and a mapping σ that maps every face F of $\mathcal{P}$ onto a set of faces (or edges or vertices) of $\mathcal{Q}$ such that there exists a continuous motion that moves every face F into $\sigma(F)$ such that (i) the L_∞ Hausdorff distance between F and its image during the motion never exceeds $\frac{3}{2}$ and (ii) if two points on two faces become equal during the motion, they remain equal through the rest of the motion.*

2.1 Main algorithm

The algorithm is organized in 4 steps. In every step of the algorithm, faces are subdivided and/or modified. We denote by $\mathcal{P}_i$ the set of faces at the end of Step i and by σ_i the mapping from the faces of $\mathcal{P}_{i-1}$ to those of $\mathcal{P}_i$ (with $\mathcal{P}_0 = \mathcal{P}$ and $\mathcal{P}_4 = \mathcal{Q}$). These mappings are trivial and not explicitly described, except in Step 1. Let $\sigma = \sigma_4 \circ \cdots \circ \sigma_1$ be the global mapping from the faces of $\mathcal{P}$ to those of the output simplicial complex $\mathcal{Q}$.

1. *Collapse the faces that are close to one another.* Order all the input faces arbitrarily from $\bar{F}_1$ to $\bar{F}_n$. During the process, we modify iteratively the faces. For clarity, we denote by F_i the faces that are iteratively modified, which we initially set to $F_i = \bar{F}_i$ for all i. Roughly speaking, for i from 1 to $n-1$, we project along y the points of $F_{i+1}, \ldots, F_n$

onto F_i but only the points that project at distance at most 1. Furthermore, we create, if needed, side walls that connect the boundary of the projected points to their pre-image. Refer to Figure 1.

a. For i from 1 to n and for j from $i+1$ to n, do
- Let R_{ji} be the polygonal region that consists of the points $p_j \in F_j$ whose projection onto F_i along the y-direction lies within distance less than 1 from p_j, i.e., $R_{ji} = \{p_j \in F_j \mid \exists\, \alpha \in (-1,1),\, \exists\, p_i \in F_i,\, p_j = p_i + \alpha\vec{j}\}$.[1]
- Modify F_j by removing R_{ji} from it.

Let $\tilde{F}_1, \ldots, \tilde{F}_n$ be the resulting faces at the end of the two nested loops and let R'_{ji} be the projection of R_{ji} on $\tilde{F}_i$ along the y-direction.

b. For j from 1 to n, consider on $\bar{F}_j$ the set of R_{ji} $(i < j)$ and consider their edges, in turn. We define new faces that connect some edges of R_{ji} and R'_{ji}, which we refer to as *connecting faces* (see e.g., faces $X_{ji\xi}$ in Figure 1). If edge e is a common edge of R_{ji} and $\tilde{F}_j$, we define a new face as the convex hull of e and its projection on F_i along y. If e is a common edge of R_{ji} and $R_{ji'}$ and if e projects (along y) on $\tilde{F}_i$ and on $\tilde{F}_{i'}$ into two distinct segments e_i and $e_{i'}$, respectively, we define a new face as the convex hull of e_i and $e_{i'}$; however, if e belongs to that face, we split it in two at e.[2]

c. For i from 1 to n, subdivide $\tilde{F}_i$ by the arrangement of edges of the R'_{ji}, $j = i+1, \ldots, n$.

To summarize, we have removed from every input face $\bar{F}_j$ the regions R_{ji}, $i = 1, \ldots, j-1$, we subdivided the resulting faces $\tilde{F}_i$ by the edges of all R'_{ji}, $j = i+1, \ldots, n$, and we created new connecting faces.

Finally, we define σ_1 to map every input face $\bar{F}_j$ to the union of the resulting face $\tilde{F}_j$, all the regions R'_{ji}, $i < j$ (subdivided as in $\tilde{F}_i$), and all the connecting faces that are defined by R_{ji}, $i < j$.

2. *Partition the space into slabs.* Project all the faces of $\mathcal{P}_1$ on the floor, compute their arrangement, lift all the resulting edges onto all faces of $\mathcal{P}_1$, and subdivide the faces accordingly; let $\mathcal{P}'_1$ be the resulting subdivision. Then, project all edges of $\mathcal{P}'_1$ on the back wall and compute their arrangement (but do not lift the resulting edges back on $\mathcal{P}'_1$).

The closed region bounded by the two side walls $x = c \pm \frac{1}{2}$, $c \in \mathbb{Z}$, is called a *thin slab* $\mathcal{S}_c$, if it contains (at least) a vertex of any of these two arrangements. A *thick slab* is a closed region bounded by two consecutive thin slabs.

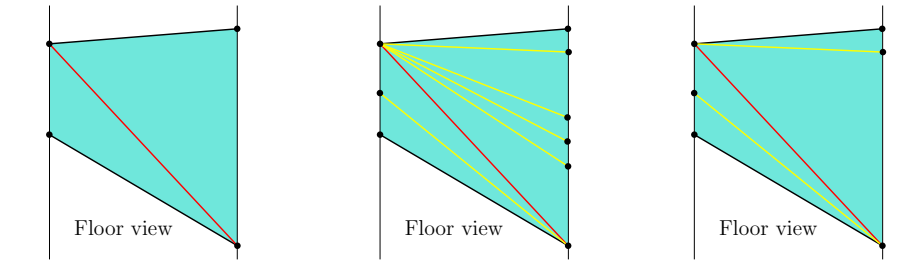
We subdivide all faces of $\mathcal{P}'_1$ by intersecting them with the side-wall boundaries of all slabs, resulting in $\mathcal{P}_2$.

Note that if two thin slabs share a side-wall plane, this plane is a thick slab between these two thin slabs. However, we treat such thick slabs as if they had infinitesimal width; their two side-wall boundaries are considered combinatorially distinct although they coincide geometrically. Thus, for instance, an edge of $\mathcal{P}'_1$ intersects such a thick slab boundary in two combinatorially distinct points that geometrically coincide.

3. *Triangulate the faces.* We triangulate all the faces of $\mathcal{P}_2$ in every slab in turn. We first consider thin slabs and then thick slabs. This order matters because, when triangulating

[1] R_{ji} is polygonal since its boundary consists of (i) segments of the boundary of F_j, (ii) segments of the boundary of F_i projected onto F_j along the y-direction, and (iii) segments of the intersection of F_j and the translated copies of F_i by vectors $\pm\vec{j}$.

[2] Connecting faces are trapezoids that intersect any side wall in segments of length at most 1 that are parallel to the y-axis. However, connecting faces may overlap and that a connecting face may not contain the edge of R_{ji} that defines it (see the full version for details).



(a) Triangulation of trapezoids not considering dummy vertices.

(b) Same as (a) but with dummy vertices.

(c) Same as (b) but on a single trapezoid.

Figure 2 Triangulations of trapezoids in a thick slab.

faces in thin slabs, we split some edges at some new vertices; when such edges and new vertices lie in the side-wall boundaries of the thin slabs, they also belong to the adjacent thick slabs and these new vertices are to be considered in these thick slabs.

a. *Thin slabs.* Project along the x-axis all the faces in a thin slab $\mathcal{S}_c$ on the side wall $x = c$ that bisects the slab, and compute the arrangement of the projected edges. In that side wall, denote as *hot* all the pixels that contain a vertex of that arrangement and split every edge that intersects a hot pixel at its intersection with the pixel boundary.[3] Triangulate the resulting arrangement[4] and lift it back (still along the x-axis) onto all the faces in the slab and subdivide them accordingly. The subdivision vertices that lie on the side-wall boundaries of $\mathcal{S}_c$ are referred to as *dummy vertices* to distinguish them from other vertices. In a thick slab of zero width, the dummy vertices typically differ on the two combinatorially-distinct though geometrically-equal side-wall boundaries.

b. *Thick slabs.* Refer to Figure 2. Not considering the dummy vertices of Step 3a, all faces are trapezoids (possibly degenerated to triangles) such that the parallel edges lie on the two side-wall boundaries of the thick slab; any two trapezoids are either identical, disjoint, or share exactly one edge or vertex, and the same holds for their projections on the floor. The dummy vertices lie on the trapezoid edges that lie on the side-wall boundaries of the thick slab.

Not considering the dummy vertices, all trapezoids that project on the floor onto one and the same trapezoid are triangulated such that all the diagonals project on the floor onto one and the same diagonal. Trapezoids can have dummy vertices only on the edges on the side walls; thus, after splitting a trapezoid in two triangles, each triangle can have dummy vertices on at most one of its edges. For every such triangle, we further triangulate it by adding an edge connecting every dummy vertex to the opposite vertex of the triangle.

4. *Snap all vertices* to the centers of their voxels. Vertices that are on the boundary of a thin slab $\mathcal{S}_c$ are snapped onto the side wall $x = c$ that bisects the slab; this is well defined even when two thin slabs share a side-wall boundary because we have considered (in Step 2) two combinatorial instances of such side walls, one associated to each of the thin

[3] As in [7], these vertices are associated with the hot pixel so that the center of the pixel they will be snapped to is well defined. This ensures that no intersection is created during the snapping motion, but simply adding one vertex on the edges and strictly inside every hot pixel yields the same result.

[4] Before triangulating, add the hot pixel boundaries to the arrangement so that the triangulating edges do not cross hot pixels. Although triangulating the faces at this stage is useful for the proof of correctness of the algorithm, it improves the complexity without changing the output to triangulate these faces at the end of the algorithm instead; see Section 2.2.

slabs. Vertices that lie on the common boundary of two voxels inside a thin slab $\mathcal{S}_c$ are associated to voxels according to the vertex-pixel associations when snap rounding in 2D the projections of the edges in $\mathcal{S}_c$ onto its bisecting side wall $x = c$.

2.2 Algorithm refinements

Subdivision of the faces in thin slabs (Step 3a). When snapping all vertices to their voxel centers in Step 4, any planar polygon in a thin slab $\mathcal{S}_c$ is transformed into a planar polygon in the side-wall $x = c$. Depending on how the vertices move toward their voxel centers, the polygon may not remain planar during the motion, but it is planar at the end of the motion. Hence, the output will be unchanged if, in Step 3a, we avoid triangulating the faces, that is, we avoid triangulating the arrangement in the side wall $x = c$ and only lift the new vertices of the arrangement onto the edges in $\mathcal{S}_c$. Still, after snapping the vertices in Step 4, the resulting polygons in the side wall $x = c$ should be triangulated so that the algorithm returns a simplicial complex.

Subdivision of the connecting faces (Steps 2, 3a and 3b). In Step 2, the connecting faces are subdivided by the side-wall boundaries of all slabs. The resulting faces are trapezoids (possibly degenerate to triangles) with two edges of length at most 1 that are parallel to the y-axis. Such trapezoids remain planar when their vertices are moved to their voxel centers in Step 4. Hence, triangulating these trapezoids does not modify the motion of any of its points. Furthermore, subdividing their edges that are parallel to the y-axis does not change the trapezoid motions either.

It follows that, in Step 2, we do not need to subdivide the connecting faces by the vertical projections of the edges of $\mathcal{P}_1$, in Step 3a, we do not need to lift any dummy vertices on the connecting-face edges that are parallel to the y-axis and in Step 3b, we do not need to triangulate the connecting faces. Still, we should triangulate the connecting faces at the end of the algorithm in order to obtain a simplicial complex which could trivially be done.

3 Proof of correctness

We prove here Theorem 1. We focus on Step 1 of the algorithm in Section 3.1, on Step 4 in Section 3.2, and we wrap up in Section 3.3.

3.1 Step 1

We first prove the main properties of Step 1 and then a technical lemma, which will be used in Lemma 6.

▶ **Lemma 2.** *Every point of the faces of $\mathcal{P}$ can be continuously moved so that every face F of $\mathcal{P}$ is continuously deformed into $\sigma_1(F)$ such that (i) the L_∞ Hausdorff distance between F and its image during the motion never exceeds 1 and (ii) if two points on two faces become equal during the motion, they remain equal through the rest of the motion.*

Sketch of proof. The complete proof is omitted for lack of space. The motion is decomposed into n successive phases, considering the projection on each F_i in turn. For a particular F_i, the naive way of moving R_{ji} to $\sigma_1(R_{ji})$ is to move R_{ji} to R'_{ji}, by moving each point of R_{ji} along the y-direction and at constant speed, and to transform edge e_ζ into face X_ζ for every edge e_ζ of the boundary of R_{ji} that defines a connecting face X_ζ. However, this does not define a function since segments are mapped to faces. The definition of a continuous

motion requires a bit of technicality but the straightforward underlying idea is to subdivide R_{ji} by considering, for each edge e_ζ, a tiny quadrilateral inside R_{ji} and bounded by e_ζ. We then transform continuously each tiny quadrilateral bounded by e_ζ into the connecting face X_ζ and move the complement of these quadrilaterals, which is a slightly shrunk version of R_{ji}, into R'_{ji}. This can be done so that when two distinct points become equal during the motion, they remain equal through the rest of the motion. The Hausdorff distance property is straightforward. ◀

▶ **Lemma 3.** *If a line L parallel to the y-axis intersects the relative interior of a face of $\mathcal{P}_1$ in a single point p then the distance along L from p to any other face of $\mathcal{P}_1$ is at least 1.*

Proof. If a line L parallel to the y-axis intersects the relative interior of a face F of $\mathcal{P}_1$ in a single point p, this face is not a connecting face. Assume for a contradiction that L intersects another face F' of $\mathcal{P}_1$ at some point p' that is at distance less than 1 from p.

Consider first the case where F' is not a connecting face. Since non-connecting faces are not parallel to the y-axis, there exists a line L' parallel to the y-axis (and close to L) that intersects the relative interiors of both F and F' in two points at distance less than 1. However, this is impossible after Step 1.

Consider now the case where F' is a connecting face. It follows from Step 1b of the algorithm that any point $p' \in F'$ lies on a segment (not necessarily entirely in F') of length at most 2, parallel to the y-axis and with its endpoints on two non-connecting faces of $\mathcal{P}_1$. Consider the shortest such segment. Unless this segment has length 2 and p is its midpoint, p is at distance less than 1 from one of the segment endpoints; considering instead of F' the non-connecting face supporting this endpoint yields a contradiction as shown above. By definition, if the segment has length 2, its midpoint p must lie on a common edge e of some R_{ji} and $R_{ji'}$ such that p projects on $\tilde{F}_i$ and on $\tilde{F}_{i'}$ into two points at distance 1 from p in the directions $\vec{\jmath}$ and $-\vec{\jmath}$, respectively. During Step 1, R_{ji} and $R_{ji'}$ are removed from the input face $\bar{F}_j$, thus the resulting face $\tilde{F}_j$ does not contain p in its interior (and not at all if p is in the interior of edge e). If the input face that contains F' is distinct from $\bar{F}_j$, then the fact that p belongs to $R_{ji} \cap R_{ji'} \subset \bar{F}_j$ and to the interior of F' contradicts the hypothesis that the input faces do not properly intersect. Otherwise, $F' \subseteq \bar{F}_j$ and thus $F' \subseteq \tilde{F}_j$, which contradicts the fact that p belongs to the interior of F' but not to the interior of $\tilde{F}_j$, and concludes the proof. ◀

3.2 Step 4

In the following, we consider in the snapping phase of Step 4 a continuous motion of the vertices such that every vertex moves on a straight line toward the center of its voxel at a speed that is constant for each vertex and so that all vertices start and end their motions simultaneously. The motion of the other points in a face move accordingly to their barycentric coordinates in the face. Note that, in every voxel that contains a vertex, the motion is a homothetic transformation whose factor goes from one to zero. During that motion, we consider that thin and thick slabs respectively shrink and expand accordingly.

We recall the standard snap-rounding result for segments in two dimensions. A pixel is called *hot* if it contains a vertex of the arrangement of segments.

▶ **Theorem 4** ([7, Thm. 1]). *Consider a set of segments in 2D split in fragments at the hot pixel boundaries and a deformation that (i) contracts homothetically all hot pixels at the same speed*[5] *and (ii) moves the fragments outside the hot pixels according to the motions of their endpoints. During the deformation, no fragment endpoint ever crosses over another fragment.*

▶ **Lemma 5.** *When moving all vertices to the center of their voxels in Step 4, no two faces, edges, or vertices of $\mathcal{P}_3$ properly intersect in thin slabs.*

Proof. Consider all the faces of $\mathcal{P}_3$ in a thin slab $\mathcal{S}_c$ and the arrangement of their projections (along the x-axis) onto the side wall $x = c$. In that side wall, a pixel that contains a vertex of the arrangement is hot and every edge (in that side wall) that intersects a hot pixel is split at the pixel boundary (Step 3a). By Theorem 4, when moving in that side-wall all the projected vertices to the centers of their pixels, the topology of the arrangement does not change except possibly at the end of the motion, where edges and vertices may become identical.

It follows that the property that every face of $\mathcal{P}_3$ in $\mathcal{S}_c$ projects onto a single face of the arrangement in the side wall $x = c$, which holds by construction at the beginning of the motion (Step 3a), holds during the whole motion of the vertices in 3D and of their projections in the side wall $x = c$.

Furthermore, the motion preserves the ordering of the x-coordinates of the vertices in $\mathcal{S}_c$, until the end when they all become equal to c. Together with the previous property, this implies that, in a thin slab, during the snapping motion, (i) no vertices and edges intersect the relative interior of a face and (ii) if two edges intersect in their relative interior, it is at the end of the motion and they become identical. Furthermore, (iii) no vertices intersect the relative interior of an edge because, in Step 3a, we have split every edge that intersects a hot pixel in projection in the side wall $x = c$. This concludes the proof. ◀

▶ **Lemma 6.** *When moving all vertices to the center of their voxels in Step 4, no two faces, edges, or vertices of $\mathcal{P}_3$ properly intersect in thick slabs.*

Proof. By construction (Step 2), all the vertices in a thick slab are on its side-wall boundaries and, in these side walls, no two edges or vertices properly intersect during the motion, by Lemma 5. Thus, we only have to consider edges that connect the two side-wall boundaries of a thick slab and show that such edges do not properly intersect during the snapping motion. Note that input faces are not vertical (i.e., not parallel to the z-axis) by assumption and connecting faces are not vertical in thick slabs since they are parallel to the y-axis and they intersect both side-wall boundaries of the thick slab.

Initially, these edges project on the floor onto edges that do not properly intersect pairwise (by definition of the slabs in Step 2). Thus, by Theorem 4, the projections on the floor of two edges either (i) coincide throughout the whole motion, or (ii) they do not properly intersect and do not coincide throughout the whole motion except possibly at the end when they may coincide. In the first case, throughout the whole motion, the edges belong to the same moving vertical plane and they do not properly intersect since they do not initially; indeed, since faces are not vertical, edges may intersect in a vertical plane only if they are boundary edges of trapezoids of $\mathcal{P}_2$, and such edges do not properly intersect on the back wall by definition of thick slabs. Hence, only in the latter case (ii), two edges may properly

[5] The proof in [7] considers separately motions in x and in y but the same argument applies for simultaneous homothetic contractions in x and y.

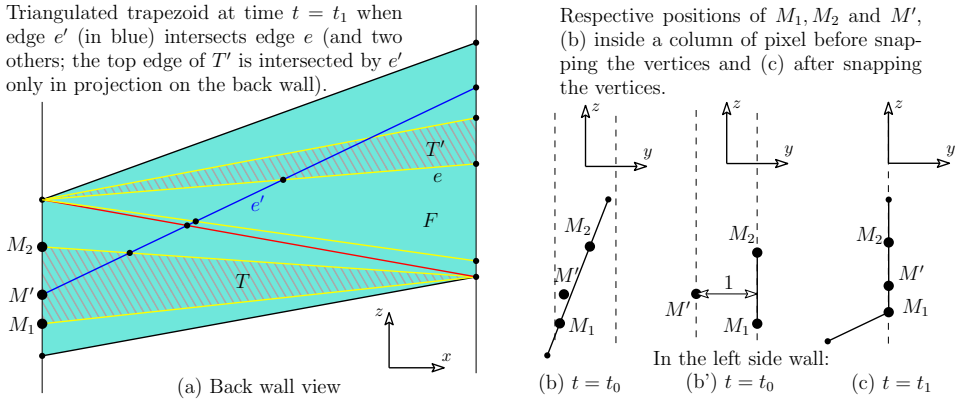


Figure 3 For the proof of Lemma 6.

intersect during the motion; furthermore, the first time this may happen is at the end of the motion and then, the two edges belong to the same vertical plane.

Similarly, applying Theorem 4 to the back-wall projection of the boundary edges of the trapezoids (but not of their triangulating edges), we get that if two boundary edges of trapezoids properly intersect in 3D during the motion, it is at the end and they must coincide in the back-wall projection. Since two edges that coincide in two projections are equal, we get that boundary edges of trapezoids cannot properly intersect throughout the motion. It remains to prove that there is no proper intersections that involve the edges triangulating the trapezoids.

Consider for a contradiction two edges e and e' that properly intersect in a vertical plane V at time t_1, the end of the motion. Since boundary edges of the trapezoids do not properly intersect, one of the two edges, say e, is a triangulation edge. Consider the trapezoid that initially contains e and its image F, at time t_1, which is a set of triangles. We prove that (at time t_1) **edge e' properly intersects at least one of the two boundary edges of F**.

Assume for a contradiction that e' properly intersects none of the two boundary edges of F and refer to Figure 3(a). Consider all the edges of the triangulation of F that are properly intersected by e' and the sequence of triangles (of that triangulation) that are incident to these edges; let T and T' denote the first and last triangles of that sequence. All these triangles except possibly one, T or T', must be in the vertical plane V; this is trivial for all triangles but T and T' and, if neither T nor T' lies in V, then edge e' properly intersects the surface formed by these triangles, contradicting the property that t_1 is the first time a proper intersection may occur.

As in Figure 3(a), assume without loss of generality that T, the bottommost triangle of the sequence, lies in the vertical plane V at time t_1. Let M' be the endpoint of e' that lies in T and let M_1M_2 be the edge of T that supports M'. At time t_1, M_1 and M_2 are vertically aligned and M' is in between them. Thus, before the snapping motion starts, at time $t = t_0$, M_1, M_2 and M' must lie in the same vertical column of pixel (in the side wall) and M' must be vertically in between M_1 and M_2 (see Figure 3(b)). However, the distance along the y-axis between M' and segment M_1M_2 is at least 1 by Lemma 3. Thus, M' and $M' \pm \vec{\jmath} \in M_1M_2$ are initially on opposite sides of the column of pixels (as in Figure 3(b')) and thus have half-integer x and y-coordinates, which contradicts item (δ) of our general position hypothesis.

Hence, at time $t = t_1$, edge e' properly intersects one of the boundary edges, say r, of trapezoid F. Since boundary edges do not properly intersect, e' must be a triangulation edge

of its trapezoid F' and we can apply the same argument as above on edges e' and r, instead of e and e'. We get that r properly intersects a boundary edge r' of F', a contradiction. ◀

3.3 Wrap up, proof of Theorem 1

First, by construction, the algorithm outputs faces that have integer coordinates.

Second, there is a continuous motion of every input face F into $\sigma(F)$ so that the Hausdorff distance between F and its image during the motion never exceeds $\frac{3}{2}$ for the L_∞ metric. Indeed, by Lemma 2, the Hausdorff distance never exceeds 1 between F and its image during the motion Step 1; in Steps 2 and 3 the faces are only subdivided; and the Hausdorff distance between any face of $\mathcal{P}_3$ and and its image during the motion of Step 4 clearly never exceeds $\frac{1}{2}$.

Third, if two points on two faces become equal during the motion, they remain equal through the rest of the motion. This is proved in Lemma 2 for the motion of Step 1 and this also holds for the motion of Step 4 since, by Lemmas 5 and 6, if two faces, edges or vertices intersect during this motion, they share a common face of both, whose motion is uniquely defined by its vertices (actually, we show in the proofs of Lemmas 5 and 6 that no two distinct points become equal except possibly at the end of the motion).

Finally, the algorithm outputs a simplicial complex by Lemmas 5 and 6.

4 Worst-case complexity

We define the z-cylinder of a face F as the volume defined by all the lines parallel to the z-axis that intersect F; similarly for x and y-cylinders. Over all input faces F, let f_d be the maximum number of input faces that are (i) intersected by one such cylinder of F and (ii) at distance at most d from F. Denote by $f = f_\infty$ the maximum number of faces intersected by one such cylinder. Let w_x be the maximum number of input faces that are intersected by any side wall $x = c$.

▶ **Proposition 7.** *Given a set of $O(n)$ polygons, each of constant complexity, the algorithm outputs a simplicial complex of complexity $O(nw_x^2 f^7 f_1^5)$ in time $O(n^2 w_x f^9 f_1^7)$.*

Proof. For analyzing the complexities of the algorithm and of its output, we bound the number of edges that we consider for splitting the input faces. However, for each face, we first count the number of edges without considering any intersection; in other words, for each face, we bound the number of lines supporting the edges instead of the complexity of the induced arrangement. To avoid confusion, we refer to these edges as unsplit edges.

Number of slabs. After projection on the back wall, the edges of R_{ji} and R'_{ji}, in Step 1, are pieces of the boundary edges of the input faces $\bar{F}_k$ and of the segments of intersection $\bar{F}_k \cap (\bar{F}_\ell \pm \vec{\jmath})$. In a y-cylinder of a face F, only f faces project on the back wall and thus there are $O(ff_1)$ such edges. Thus, at the end of Step 1, every input face ends up supporting $O(ff_1)$ unsplit edges. In Step 2, we thus lift $O(f^2 f_1)$ unsplit edges onto every face. Every unsplit edge on a given face F may only intersect, after projection on the back wall, edges that lie on the faces that intersect the y-cylinder of F; there are $O(f)$ such faces and $O(f^2 f_1)$ unsplit edges on each of them, thus every unsplit edge may intersect $O(f^3 f_1)$ edges on the back wall. There are $O(nf^2 f_1)$ unsplit edges in total, hence, in Step 2, the back wall arrangement has complexity $O(nf^5 f_1^2)$. (Similarly, the floor arrangement has complexity $O(nf^3 f_1^2)$.) The number of thin and thick slabs is thus $O(nf^5 f_1^2)$.

Complexity in thin slabs. For any thin slab $\mathcal{S}_c$, we analyze the complexity of the output in the side wall $x = c$. We do not consider here any triangulation edge inside the faces as discussed in Section 2.2. Thus, the edges in $\mathcal{S}_c$ are subdivisions of the edges of $\mathcal{P}_1$ and subdivisions of the edges defined as the intersection of the faces of $\mathcal{P}_1$ and the side walls $x = c \pm \frac{1}{2}$. More precisely, these edges are pieces of either (a.i) edges of $\mathcal{P}_1$ that lie on the input faces, (a.ii) edges of intersection of an input face with a side wall $x = c \pm \frac{1}{2}$, (b.i) edges of connecting faces that are parallel to the y-axis, or (b.ii) edges of intersection of a connecting face with a side wall $x = c \pm \frac{1}{2}$. Note that, although we do not consider the faces triangulated in $\mathcal{S}_c$, the edges are subdivided according to the arrangement of their projections on the side wall $x = c$; however, we consider them unsplit for now.

As mentioned above, every input face supports $O(ff_1)$ unsplit edges at the end of Step 1. Thus, if $F_{\mathcal{S}_c}$ denotes the number of input faces that are intersected by the thin slab $\mathcal{S}_c$, there are $O(F_{\mathcal{S}_c} ff_1)$ edges of types (a) in $\mathcal{S}_c$. On the other hand, the $O(ff_1)$ unsplit edges on the back wall yield an arrangement of size $O(f^2 f_1^2)$; viewed on the back wall, edges of type (b) are incident to the vertices of this arrangement, so the number of edges of type (b) in $\mathcal{S}_c$ is bounded by $O(F_{\mathcal{S}_c} f^2 f_1^2)$.

We bound the number of intersections between these edges at the end of the algorithm. As noted in Section 2.2, we do not consider intersections that involve edges of type (b) because such intersections necessarily belong to the hot pixels defined by the edge endpoints. We thus consider here only edges of type (a). Every edge on a given input face F may only intersect, after projection on the side wall $x = c$, edges that lie on faces that intersect $\mathcal{S}_c$ and the x-cylinder of F; there are $O(f_1)$ such faces and $O(ff_1)$ edges on each of them, thus every edge may intersect $O(ff_1^2)$ edges on the side wall $x = c$. There are $O(F_{\mathcal{S}_c} ff_1)$ edges of type (a) in $\mathcal{S}_c$, thus, there are $O(F_{\mathcal{S}_c} f^2 f_1^3)$ intersections between edges of type (a).

In addition to these $O(F_{\mathcal{S}_c} f^2 f_1^3)$ intersections, there are $O(F_{\mathcal{S}_c} f^2 f_1^2)$ edges in $\mathcal{S}_c$. The number of hot pixels in the side wall $x = c$ is thus $H_c = O(F_{\mathcal{S}_c} f^2 f_1^3)$ at the end of Step 3a.

The number of vertices (dummy or not) in $\mathcal{S}_c$ at the end of Step 3a is the number of hot pixels, H_c, times the number of edges in $\mathcal{S}_c$, $O(F_{\mathcal{S}_c} f^2 f_1^2)$. However, the size of the arrangement in the side wall $x = c$ after snapping the vertices to their voxel centers in Step 4 is of the order of the number of hot pixels, H_c, and it remains as such after triangulating the faces (see Section 2.2).

Complexity in thick slabs. The number of edges in a thick slab after the triangulation of Step 3b is (i) the number of edges of connecting faces in the slab, that is $O(w_x ff_1)$ (the number $O(ff_1)$ of unsplit edges on input faces times the number $O(w_x)$ of input faces that intersect the slab) plus (ii) the number $O(w_x)$ of input faces that intersect the slab times the number $H_{c_1} + H_{c_2}$ of hot pixels in the adjacent thin slabs $\mathcal{S}_{c_1}$ and $\mathcal{S}_{c_2}$. This sums up to $O(w_x(H_{c_1} + H_{c_2}))$.

Complexity of the output. The total complexity of the output is thus $O(w_x)$ times the sum over all thin slabs of the number of hot pixels H_c in $\mathcal{S}_c$. Denoting by $\mathring{F}_{\mathcal{S}_c}$ the number of input faces that are entirely inside $\mathcal{S}_c$, we have that $F_{\mathcal{S}_c} \leqslant \mathring{F}_{\mathcal{S}_c} + 2w_x$ and that the sum over all thin slabs of $\mathring{F}_{\mathcal{S}_c}$ is $O(n)$. Hence, the sum over all $N = O(nf^5 f_1^2)$ thin slabs of the numbers of hot pixels $H_c = O(F_{\mathcal{S}_c} f^2 f_1^3)$ is $O((n + Nw_x) f^2 f_1^3) = O(nw_x f^7 f_1^5)$. Times w_x gives the complexity of the output: $O(nw_x^2 f^7 f_1^5)$.

Time complexity. The time complexity is straightforward given the above analysis and the observation that the complexity of $\mathcal{P}_3$ can be larger than the rounded output. ◀

5 Complexity under some assumptions

We consider, in the following proposition, the complexity of our algorithm for approximations of "nice" surfaces, defined as follows.

▶ **Definition 8.** An (ε, κ)-sampling of a surface $\mathcal{S}$ is a set of vertices on $\mathcal{S}$ so that there is at least 1 and at most κ vertices strictly inside any ball of radius ε centered on $\mathcal{S}$. It is straightforward that a (ε, κ)-sampling of a fixed compact surface has $\Theta(n)$ vertices with $n = \frac{1}{\varepsilon^2}$ (the constant hidden in the Θ complexity depends on the area of the surface).

▶ **Definition 9.** The Delaunay triangulation of a set of points $\mathcal{P}$ restricted to a surface $\mathcal{S}$ is the set of simplices of the Delaunay triangulation of $\mathcal{P}$ whose dual Voronoi faces intersect $\mathcal{S}$. If $\mathcal{P} \subset \mathcal{S}$, we simply refer to the restricted Delaunay triangulation of $\mathcal{P}$ on $\mathcal{S}$.

▶ **Definition 10** (Nice surfaces)**.** A surface $\mathcal{S}$ is k-monotone (with respect to z) if every line parallel to the z-axis intersects $\mathcal{S}$ in at most k points. Let Δ and k be any two positive constants. A surface $\mathcal{S}$ is *nice* if it is a compact smooth k-monotone surface such that the Gaussian curvature of $\mathcal{S}$ is larger than a positive constant in a ball of radius Δ centered at any point $p \in \mathcal{S}$ where the tangent plane to $\mathcal{S}$ is vertical.

For instance, a compact smooth algebraic surface whose silhouette (with respect to the vertical direction) is a single convex curve is nice for suitable choices of Δ and k.

▶ Remark 11. The following complexities are asymptotic when n goes to infinity (or ε to zero) with hidden constants depending on the surface areas, Δ, and k. It is important to notice that these complexities are independent from the voxel size, which can go to zero with no changes in the complexities. Of course if the grid size and the surface are fixed, the total number of voxels intersecting the surface is constant and so is the size of a rounding.

The following lemma is a technical though rather straightforward result, whose proof is omitted for lack of space.

▶ **Lemma 12.** *The restricted Delaunay triangulation $\mathcal{T}$ of a (ε, κ)-sampling of a nice surface has complexity $O(n) = O(\frac{1}{\varepsilon^2})$. Any plane $x = c$ intersects at most $O(\sqrt{n}) = O(\frac{1}{\varepsilon})$ faces of $\mathcal{T}$. Furthermore, for any face f of $\mathcal{T}$, the set of vertical lines through f intersects at most $O(n^{\frac{1}{4}}) = O(\frac{1}{\sqrt{\varepsilon}})$ faces of $\mathcal{T}$.*

▶ **Proposition 13.** *Given the arrangement of the restricted Delaunay triangulations of the (ε, κ)-samplings of a constant number of nice surfaces, the algorithm outputs a simplicial complex of complexity $O(n^5)$ in time $O(n^6\sqrt{n})$.*

Proof. Consider the restricted Delaunay triangulations of the (ε, κ)-samplings of two surfaces. By definition of (ε, κ)-samplings, it is straightforward that any triangle of one triangulation intersects a constant number of triangles of the other triangulation. Hence, the complexities of Lemma 12 hold for the arrangement of the two triangulations, and similarly for a constant number of triangulations. Thus, for the arrangement of triangulations, Lemma 12 yields $w_x = O(\sqrt{n})$ and $f_1 < f = O(\sqrt[4]{n})$ (as defined in Section 4) and plugging these values in the complexities of Proposition 7 yields the result. ◀

▶ Remark 14. In practice on realistic data, one can anticipate better time and space complexities of $O(n\sqrt{n})$. Indeed, $f_1 < f$ should behave as if they were in $O(1)$ and in 2D arrangements, hot pixels are usually intersected by $O(1)$ segments. Then, with $w_x = O(\sqrt{n})$ as in Lemma 12, the proof of Proposition 7 yields complexities in $O(n\sqrt{n})$.

References

1. Mark de Berg, Dan Halperin, and Mark Overmars. An intersection-sensitive algorithm for snap rounding. *Computational Geometry*, 36(3):159–165, 2007. doi:10.1016/j.comgeo.2006.03.002.
2. Olivier Devillers, Menelaos Karavelas, and Monique Teillaud. Qualitative Symbolic Perturbation: Two Applications of a New Geometry-based Perturbation Framework. *Journal of Computational Geometry*, 8(1):282–315, 2017. doi:10.20382/jocg.v8i1a11.
3. Steven Fortune. Polyhedral modelling with multiprecision integer arithmetic. *Computer-Aided Design*, 29(2):123–133, 1997. Solid Modelling. doi:10.1016/S0010-4485(96)00041-3.
4. Steven Fortune. Vertex-rounding a three-dimensional polyhedral subdivision. *Discrete & Computational Geometry*, 22(4):593–618, 1999. doi:10.1007/PL00009480.
5. Michael T. Goodrich, Leonidas J. Guibas, John Hershberger, and Paul J. Tanenbaum. Snap rounding line segments efficiently in two and three dimensions. In *Proceedings of the thirteenth annual symposium on Computational geometry*, pages 284–293. ACM, 1997. doi:10.1145/262839.262985.
6. Daniel H. Greene and F. Frances Yao. Finite-resolution computational geometry. In *Foundations of Computer Science, 1986., 27th Annual Symposium on*, pages 143–152. IEEE, 1986. doi:10.1109/SFCS.1986.19.
7. Leonidas J. Guibas and David H. Marimont. Rounding arrangements dynamically. *International Journal of Computational Geometry & Applications*, 8(02):157–178, 1998. doi:10.1142/S0218195998000096.
8. Dan Halperin and Eli Packer. Iterated snap rounding. *Computational Geometry*, 23(2):209–225, 2002. doi:10.1016/S0925-7721(01)00064-5.
9. John Hershberger. Improved output-sensitive snap rounding. *Discrete & Computational Geometry*, 39(1):298–318, Mar 2008. doi:10.1007/s00454-007-9015-0.
10. John Hershberger. Stable snap rounding. *Computational Geometry*, 46(4):403–416, 2013. doi:10.1016/j.comgeo.2012.02.011.
11. John D. Hobby. Practical segment intersection with finite precision output. *Computational Geometry*, 13(4):199–214, 1999. doi:10.1016/S0925-7721(99)00021-8.
12. V. Milenkovic and L. R. Nackman. Finding compact coordinate representations for polygons and polyhedra. *IBM Journal of Research and Development*, 34(35):753–769, 1990.
13. Victor Milenkovic. Rounding face lattices in d dimensions. In *Proceedings of the 2nd Canadian Conference on Computational geometry*, pages 40–45, 1990.
14. Eli Packer. Iterated snap rounding with bounded drift. *Computational Geometry*, 40(3):231–251, 2008. doi:10.1016/j.comgeo.2007.09.002.

Graph Reconstruction by Discrete Morse Theory

Tamal K. Dey
The Ohio State University, Columbus, OH, USA
tamaldey@cse.ohio-state.edu

Jiayuan Wang
The Ohio State University, Columbus, OH, USA
wang.6195@buckeyemail.osu.edu

Yusu Wang
The Ohio State University, Columbus, OH, USA
yusu@cse.ohio-state.edu

Abstract

Recovering hidden graph-like structures from potentially noisy data is a fundamental task in modern data analysis. Recently, a persistence-guided discrete Morse-based framework to extract a geometric graph from low-dimensional data has become popular. However, to date, there is very limited theoretical understanding of this framework in terms of graph reconstruction. This paper makes a first step towards closing this gap. Specifically, first, leveraging existing theoretical understanding of persistence-guided discrete Morse cancellation, we provide a simplified version of the existing discrete Morse-based graph reconstruction algorithm. We then introduce a simple and natural noise model and show that the aforementioned framework can correctly reconstruct a graph under this noise model, in the sense that it has the same loop structure as the hidden ground-truth graph, and is also geometrically close. We also provide some experimental results for our simplified graph-reconstruction algorithm.

2012 ACM Subject Classification Mathematics of computing → Graph theory, Theory of computation → Computational geometry, Computing methodologies → Shape modeling

Keywords and phrases graph reconstruction, discrete Morse theory, persistence

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.31

Related Version A full version of this paper is available at `https://arxiv.org/abs/1803.05093`

Funding This work is supported by National Science Foundation under grants CCF-1526513, CCF-1618247, and CCF-1740761, and by National Institute of Health under grant R01EB022899.

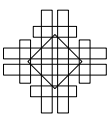
Acknowledgements We thank Suyi Wang for generously helping with the software. The Enzo dataset used in our experiments is obtained from [12]. The Bone dataset is obtained from [13].

1 Introduction

Recovering hidden structures from potentially noisy data is a fundamental task in modern data analysis. A particular type of structure often of interest is the geometric graph-like structure. For example, given a collection of GPS trajectories, recovering the hidden road network can be modeled as reconstructing a geometric graph embedded in the plane. Given the simulated density field of dark matters in universe, finding the hidden filamentary structures is essentially a problem of geometric graph reconstruction.

Different approaches have been developed for reconstructing a curve or a metric graph from input data. For example, in computer graphics, much work have been done in extracting

34th International Symposium on Computational Geometry (SoCG 2018).
Editors: Bettina Speckmann and Csaba D. Tóth; Article No. 31; pp. 31:1–31:15



Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1D skeleton of geometric models using the medial axis or Reeb graphs [7, 26, 19, 15, 21, 4]. In computer vision and machine learning, a series of work has been developed based on the concept of *principal curves*, originally proposed by Hastie and Steutzle [17]. Extensions to graphs include the work in [18] for 2D images and in [22] for high dimensional point data.

In general, there is little theoretical guarantees for most approaches developed in practice to extract hidden graphs. One exception is some recent work in computational topology: Aanijaneya et al. [1] proposed the first algorithm to approximate a metric graph from an input metric space with guarantees. The authors of [5, 15] used Reeb-like structures to approximate a hidden (metric) graph with some theoretical guarantees. These work however only handles (Gromov-)Hausdorff-type of noise. When input points are embedded in an ambient space, they requires the input points to lie within a small tubular neighborhood of the hidden graph. Empirically, these methods do not seem to be effective when the input contains ambient noise allowing some faraway points from the hidden graph.

Recently, a discrete Morse-based framework for recovering hidden structures was proposed and studied [6, 16, 23]. This line of work computes and simplifies a discrete analog of (un)stable manifolds of a Morse function by using the (Forman's) discrete Morse theory coupled with persistent homology for 2D or 3D volumetric data. One of the main issues in such simplification is the inherent obstructions that may occur for cancelling critical pairs. The authors of [23] suggest sidestepping this and consider a combinatorial representation of critical pairs for further processing. The authors in [6] identify a restricted set of pairs called "cancellable close pairs" which are guaranteed to admit cancellation. This framework has been applied to, for example, extracting filament structures from simulated dark matter density fields [24] and reconstructing road networks from GPS traces [25].

This persistence-guided discrete Morse-based framework has shown to be very effective in recovering a hidden geometric graph from (non-Hausdorff type) noise and non-homogeneous data. The method draws upon the global topological structure hidden in the input scalar field and thus is particularly effective at identifying junction nodes which has been a challenge for previous approaches that rely mostly on local information. However, to date, theoretical understanding of such a framework remains limited. Simplification of a discrete Morse gradient vector field using persistence has been studied before. For example, the work of [6] clarifies the connection between persistence-pairing and the simplification of *discrete Morse chain complex* (which is closely related, but different from the cancellation in the discrete gradient vector field) for 2D and 3D domains. Bauer et al. [3] obtain several results on persistence guided discrete Morse simplification for combinatorial surfaces. The simplification of vertex-edge persistence pairing used in [3] has also been observed in [2] independently for simplifying Morse functions on surfaces. Leveraging these existing developments, we aim to provide a theoretical understanding of a persistence-guided discrete Morse based approach to reconstruct a hidden geometric graph.

Main contributions and organization of paper. In Section 3, we start with one version of the existing persistence-guided discrete Morse-based graph reconstruction algorithm (as employed in [24, 25, 8]). We show that this algorithm can be significantly simplified while still yielding the same output. To establish the theoretical guarantee of the reconstruction algorithm, we introduce a simple yet natural noise model in Section 4. Intuitively, this noise model assumes that we are given an input density field $\rho : \mathbb{R}^d \to \mathbb{R}$ where densities are significantly higher within a small neighborhood around a hidden graph than outside it. Under this noise model, we show that the reconstructed graph has the same loop structure as the hidden graph, and is also geometrically close to it; the technical details are in Sections 5 and

6 for the general case and the 2-dimensional case (with additional guarantees), respectively.

While our noise model is simple, our theoretical guarantees are first of a kind developed for a discrete Morse-based approach applied to graph reconstruction. In fact, prior to this, it was not clear whether a discrete Morse based approach can recover a graph even if there is no noise, that is, the density function has positive values *only* on the hidden graph. For our specific noise model, it may be possible to develop thresholding strategies perhaps with theoretical guarantees. However, previous work (e.g, [24, 25]) have shown that discrete Morse approach succeeds in many cases handling non-homogeneous data where thresholding fails. We have implemented the proposed simplified algorithm and tested it on several data sets, which generally gives a speed-up of at least a factor of 2 over a state-of-the-art approach. We present more discussions and experimental results in the full version of this paper [9].

2 Preliminaries

2.1 Morse theory

For simplicity, we consider only a smooth function $f : \mathbb{R}^d \to \mathbb{R}$. See [10, 20] for more general discussions.

For a point $p \in \mathbb{R}^d$, the gradient vector of f at a point p is $\nabla f(p) = -[\frac{\partial f}{\partial x_1} \cdots \frac{\partial f}{\partial x_d}]^T$, which represents the steepest descending direction of f at p, with its magnitude being the rate of change. An integral line of f is a path $\pi : (0, 1) \to \mathbb{R}^d$ such that the tangent vector at each point p of this path equals $\nabla f(p)$, which is intuitively a flow line following the steepest descending direction at any point. A point $p \in \mathbb{R}^d$ is *critical* if its gradient vector vanishes, i.e, $\nabla f(p) = [0 \cdots 0]^T$. A *maximal* integral line necessarily "starts" and "ends" at critical points of f; that is, $\lim_{t\to 0} \pi(t) = p$ with $\nabla f(p) = [0 \cdots 0]^T$, and $\lim_{t\to 1} \pi(t) = q$ with $\nabla f(q) = [0 \cdots 0]^T$. See Figure 1a where we show the graph of a function $f : \mathbb{R}^2 \to \mathbb{R}$, and there is an integral line from p' to the minimum v_1.

For a critical point p, the union of p and all the points from integral lines flowing into p is referred to as the *stable manifold of* p. Similarly, for a critical point q, the union of q and all the points on integral lines starting from q is called the *unstable manifold of* q. The stable manifold of a minimum p intuitively corresponds the basin/valley around p in the terrain of f. The 1-stable manifolds of index $(d-1)$ saddles consist of pieces of curves connecting $(d-1)$-saddles to maxima – These curves intuitively capture "mountain ridges" of the terrain (graph of the function f); see Figure 1a for an example. Symmetrically, the unstable manifold of a maximum q corresponds to the mountain around q. The 1-unstable manifolds consist of a collection of curves connecting 1-saddles to minima, corresponding intuitively to the "valley ridges".

In this paper, we focus on a graph-reconstruction framework using Morse-theory (as in e.g, [16, 6, 24, 25]). Intuitively, the 1-stable manifolds of saddles (mountain ridges) of the density field ρ are used to capture the hidden graphs. To implement such an idea in practice, the *discrete Morse theory* is used for robustness and simplicity contributed by its combinatorial nature; and a simplification scheme guided by the persistence pairings is employed to remove noise. Below, we introduce some necessary background notions in these topics.

2.2 Discrete Morse theory

First we briefly describe some notions from discrete Morse theory (originally introduced by Forman [14]) in the simplicial setting.

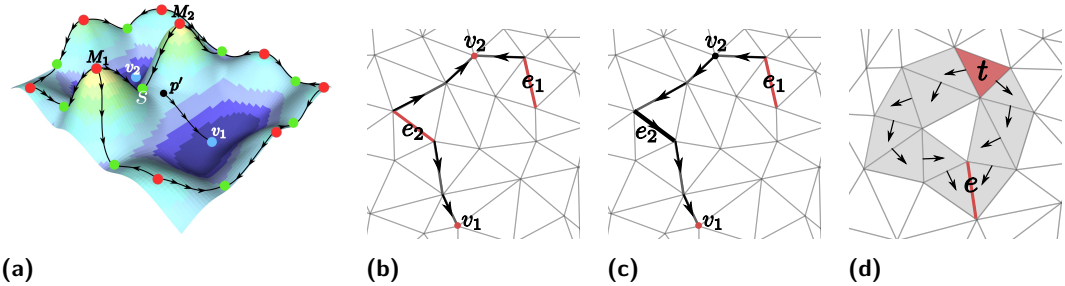


Figure 1 (a) M_1 and M_2 are maxima (red dots), v_1 and v_2 are minima (blue dots), s is a saddle (green dots) with its stable manifolds flowing to it from M_1 and M_2. If we put a drop of water at p' it will flow to v_1. If we put it on the other side of the mountain ridge it will flow to minimum v_2. (b) Before cancellation of pair $\langle v_2, e_2 \rangle$. (c) After cancellation, the path from e_2 to v_2 is inverted, giving rise to a gradient path from e_1 to v_1, making $\langle v_1, e_1 \rangle$ now potentially cancellable. (d) An edge-triangle pair $\langle e, t \rangle$ which is not cancellable as there are two gradient paths between them.

A k-simplex $\tau = \{p_0, \ldots, p_k\}$ is the convex hull of $k+1$ affinely independent points; k is called the *dimension* of τ. A *face* σ *of* τ is a simplex spanned by a proper subset of vertices of τ; σ is a *facet* of the k-simplex τ, denoted by $\sigma < \tau$, if its dimension is $k-1$.

Suppose we are given a simplicial complex K which is simply a collection of simplices and all their faces so that if two simplices intersect, they do so in a common face. A *discrete (gradient) vector* is a pair of simplices (σ, τ) such that $\sigma < \tau$. A *Morse pairing* in K is a collection of discrete vectors $M(K) = \{(\sigma, \tau)\}$ where each simplex appears in *at most* one pair; simplices that are not in any pair are called *critical*.

Given a Morse pairing $M(K)$, a *V-path* is a sequence $\tau_0, \sigma_1, \tau_1, \ldots, \sigma_\ell, \tau_\ell, \sigma_{\ell+1}$, where $(\sigma_i, \tau_i) \in M(K)$ for every $i = 1, \ldots, \ell$, and each σ_{i+1} is a facet of τ_i for each $i = 0, \ldots, \ell$. If $\ell = 0$, the V-path is *trivial*. This V-path is *cyclic* if $\ell > 0$ and $(\sigma_{\ell+1}, \tau_0) \in M(K)$; otherwise, it is *acyclic* in which case we call this V-path a *gradient path*. We say that a gradient path is a vertex-edge gradient path if $dimension(\sigma_i) = 0$, implying that $dimension(\tau_i) = 1$. Similarly, it is a edge-triangle gradient path if $dimension(\sigma_i) = 1$. A Morse pairing $M(K)$ becomes a *discrete gradient vector field* (or equivalently a *gradient Morse pairing*) if there is no cyclic V-path induced by $M(K)$.

Intuitively, given a discrete gradient vector field $M(K)$, a gradient path $\tau_0, \sigma_1, \ldots, \tau_\ell, \sigma_{\ell+1}$ is the analog of an integral line in the smooth setting. But different from the smooth setting, a maximal gradient path may not start or end at critical simplices. However, those that do (i.e, when τ_0 and σ_{k+1} are critical simplices) are analogous to maximal integral line in the smooth setting which "start" and "end" at critical points, and for convenience one can think of *critical k-simplices* in the discrete Morse setting as *index-k critical points* in the smooth setting. For example, for a function on $\mathbb{R}^2$, critical 0-, 1- and 2-simplices in the discrete Morse setting correspond to minima, saddles and maxima in the smooth setting, respectively.

For a critical edge e, we define its *stable manifold* to be the union of edge-triangle gradient paths that ends at e. Its *unstable manifold* is defined to be the union of vertex-edge gradient paths that begins with e. While earlier we use "mountain ridges" (1-stable manifolds) to motivate the graph reconstruction framework, algorithmically (especially for the Morse cancellations below), vertex-edge gradient paths are simpler to handle. Hence in our algorithm below, we in fact consider the function $g_\rho = -\rho$ (instead of the density field ρ itself) and the algorithm outputs (a subset of) the *1-unstable manifolds* (vertex-edge paths in the discrete setting) as the recovered hidden graph.

Morse cancellation / simplification. One can simplify a discrete gradient vector field $M(K)$ (i.e, reducing the number of critical simplices) by the following Morse cancellation operation: A pair of critical simplices $\langle \sigma, \tau \rangle$ with $\mathit{dimension}(\tau) = \mathit{dimension}(\sigma) + 1$ is *cancellable*, if there is a *unique* gradient path $\tau = \tau_0, \sigma_1, \ldots, \tau_\ell, \sigma_{\ell+1} = \sigma$ starting at the $k+1$-simplex τ and ends at the k-simplex σ. The *Morse cancellation operation on* $\langle \sigma, \tau \rangle$ then modifies the vector field $M(K)$ by removing all gradient vectors (σ_i, τ_i), for $i = 1, \ldots, \ell$, while adding new gradient vectors (σ_i, τ_{i-1}), for $i = 1, \ldots, \ell+1$. Intuitively, the gradient path is inverted. Note that $\tau = \tau_0$ and $\sigma = \sigma_{\ell+1}$ are no longer critical after the cancellation as they now participate in discrete gradient vectors. If there is no gradient path, or more than one gradient path between this pair of critical simplices $\langle \sigma, \tau \rangle$, then this pair is *not cancellable* – the uniqueness condition is to ensure that no cyclic V-paths are formed after the cancellation operation. See Figure 1 (b) – (d) for examples.

2.3 Persistence pairing

The Morse cancellation can be applied to any sequence of critical simplices pairs as long as they are cancellable at the time of cancellation. There is no canonical cancellation sequence. To cancel features corresponding to "noise" w.r.t. an input piecewise-linear function $f : |K| \to \mathbb{R}$, a popular strategy is to guide the Morse cancellation by the persistent homology induced by the lower-star filtration [16, 24], which we introduce now.

Filtrations and lower-star filtration. Given a simplicial complex K, let S be an ordered sequence $\sigma_1, \ldots, \sigma_N$ of all n simplices in K so that for any simplex $\sigma_i \in K$, all of its faces appear before it in S. Then S induces a *(simplex-wise) filtration* $F(K)$*:* $K_1 \subset K_2 \subset \cdots \subset K_N = K$, where $K_i = \bigcup_{j \le i} \sigma_j$ is the subcomplex formed by the prefix $\sigma_1, \ldots, \sigma_i$ of S. Passing to homology groups, we have a *persistence module* $H_*(K_1) \to \cdots \to H_*(K_N)$, which has a unique decomposition into the direct sum of a set of indecomposable summands that can be represented by the set of *persistence-pairing* $P(K)$ *induced by* $F(K)$: Each *persistence pair* $(\sigma_i, \sigma_j) \in P(K)$ indicates that a new k-th homological class, $k = \mathit{dimension}(\sigma_i)$, is created at K_i and destroyed at K_j; σ_i is thus called a *positive simplex* as it creates, and σ_j a *negative simplex*. Assuming that there is a *simplex-wise function* $\bar{f} : K \to \mathbb{R}$ such that $\bar{f}(\sigma_i) \le \bar{f}(\sigma_j)$ if $i < j$, then the *persistence* of the pair (σ, τ) is defined as $\mathrm{pers}(\sigma) = \mathrm{pers}(\tau) = \mathrm{pers}(\sigma, \tau) = \bar{f}(\tau) - \bar{f}(\sigma)$. Some simplices σ_ℓ's may be unpaired, meaning that homological features created at K_ℓ are never destroyed. We augment $P(K)$ by adding (σ_ℓ, ∞) for every unpaired simplex σ_ℓ to it, and set $\mathrm{pers}(\sigma_\ell, \infty) = \infty$.

The persistent homology can be defined for any filtration of K. In our setting, there is an input function $f : V(K) \to \mathbb{R}$ defined at the vertices $V(K)$ of K whose linear extension leads to a piecewise-linear (PL) function still denoted by $f : |K| \to \mathbb{R}$. To reflect topological features of f, we use the lower-star filtration of K induced by f: Specifically, for any vertex $v \in V(K)$, its lower-star $\mathrm{LowSt}(v)$ is the set of simplicies containing v where v has the highest f value among their vertices. Now sort vertices of K in non-decreasing order of their f-values: $v_1, \ldots, v_n$. An ordered sequence $S = \langle \sigma_1, \ldots, \sigma_N \rangle$ induces a *lower-star filtration* $F_f(K)$ *of* K *w.r.t.* f if S can be partitioned to n consecutive pieces $\langle \sigma_1, \ldots, \sigma_{I_1} \rangle$, $\langle \sigma_{I_1+1}, \ldots, \sigma_{I_2} \rangle$, $\ldots, \langle \sigma_{I_{n-1}+1}, \ldots, \sigma_N \rangle$, such that the i-th piece $\langle \sigma_{I_{i-1}+1}, \ldots, \sigma_{I_i} \rangle$ equals $\mathrm{LowSt}(v_i)$.

Now let $P_f(K)$ be the resulting set of persistence pairs induced by the lower-star filtration $F_f(K)$. Extend the function $f : V(K) \to \mathbb{R}$ to a simplex-wise function $\bar{f} : K \to \mathbb{R}$ where $\bar{f}(\sigma) = \max_{v \in \sigma} f(v)$ (i.e, $\bar{f}(\sigma)$ is the highest f-value of any of its vertices). For each pair (σ, τ), we measure its persistence by $\mathrm{pers}(\sigma, \tau) = \bar{f}(\tau) - \bar{f}(\sigma)$. Every simplex in K contributes to a persistence pair in $P_f(K)$. However, assuming the value of f is distinct on all vertices,

then those persistence pairs with zero-persistence are "trivial" in the sense they correspond to the local pairing of two simplices from the lower-star of the same vertex. A persistence pair (σ, τ) *with positive persistence* corresponds to a pair of (homological) critical points (p, q) for the PL-function $f : |K| \to \mathbb{R}$ [10] induced by the function f on $V(K)$, with $p \in \sigma$ and $q \in \tau$.

3 Reconstruction algorithm

Problem setup. Suppose we have a domain Ω (which will be a cube in $\mathbb{R}^d$ in this paper) and a density function $\rho : \Omega \to \mathbb{R}$ (that "concentrates" around a hidden geometric graph $G \subset \Omega$). In the discrete setting, our input will be a triangulation K of Ω and a density function given as a PL-function $\rho : K \to \mathbb{R}$. Our goal is to compute a graph $\widehat{G}$ approximating the hidden graph G. In Algorithm 1, we first present a *known* discrete Morse-based graph (1-skeleton) reconstruction framework, which is based on the approaches in [16, 6, 24, 25].

Algorithm 1: MorseRecon(K,ρ, δ)

Data: Triangulation K of Ω, density function $\rho : K \to \mathbb{R}$, threshold δ
Result: Reconstructed graph $\widehat{G}$
begin
1. Compute persistence pairings $P(K)$ by the lower-star filtration of K w.r.t $g_\rho = -\rho$
2. $M =$PerSimpVF$(P(K), \delta)$
3. $\widehat{G} =$ CollectOutputG(M)
4. **return** $\widehat{G}$

Procedure PerSimpVF*$(P(K), \delta)$*
1. Set initial discrete gradient field M on K to be trivial
2. Rank all persistence pairs in $P(K)$ in increasing order of their persistence
3. **for** *each* $(\sigma, \tau) \in P(K)$ *with* $\text{pers}(\sigma, \tau) \leq \delta$ **do**
4. If possible, perform discrete-Morse cancellation of (σ, τ) and update the discrete gradient vector field M
5. **return** M

Procedure CollectOutputG*(M)*
1. $\widehat{G} = \emptyset$
2. **for** *each remaining critical edge* e *with* $\text{pers}(e) > \delta$ **do**
3. $\widehat{G} = \widehat{G} \bigcup \{1\text{-unstable manifold of } e\}$
4. **return** $\widehat{G}$

Intuitively, we wish to use "mountain ridges" of the density field to approximate the hidden graph, which are computed as the 1-unstable manifolds of $g_\rho = -\rho$, the negation of the density function. Specifically, after initializing the discrete gradient vector field M to be a trivial one, a persistence-guided Morse cancellation step is performed in Procedure PerSimpVF() to compute a new discrete gradient vector field M_δ so as to capture only important (high persistent) features of g_ρ. In particular, Morse-cancellation is performed for each pair of critical simplices from $P(K)$ (if possible) in increasing order of persistence values (for pairs with equal persistence, we use the nested order as in [3]). Finally, the union of the 1-unstable manifolds of all remaining high-persistence critical edges is taken as the output graph $\widehat{G}$, as outlined in Procedure CollectOutputG().

Algorithm 2: MorseReconSimp(K,ρ, δ)

Procedure PerSimpTree$(P(K), \delta)$ /* *This procedure replaces original* PerSimpVF() */

$\Pi :=$ the set of vertex-edge persistence pairs from $P(K)$
Set $\Pi_{\leq\delta} \subseteq \Pi$ to be $\Pi_{\leq\delta} = \{(v,e) \in \Pi \mid \text{pers}(v,e) \leq \delta\}$
$\mathcal{T} := \bigcup_{(v,\sigma)\in\Pi_{\leq\delta}} \{\sigma = \langle u_1, u_2\rangle, u_1, u_2\}$
return $\mathcal{T}$

Procedure Treebased-OutputG$(\mathcal{T})$ /* *This procedure replaces* CollectOutputG() */

$\widehat{G} = \emptyset$
for *each edge* $e = \langle u, v\rangle$ *with* $\text{pers}(e) > \delta$ **do**
  Let $\pi(u)$ be the unique path from u to the sink of the tree T_i containing u
  Define $\pi(v)$ similarly; Set $\widehat{G} = \widehat{G} \cup \pi(u) \cup \pi(v) \cup \{e\}$
return $\widehat{G}$

Since we only need 1-unstable manifolds, K is assumed to be a 2-complex. It is pointed out in [8] that in fact, instead of performing Morse-cancellation for all critical pairs involving edges (i.e, vertex-edge pairs and edge-triangle pairs), one only needs to cancel vertex-edge pairs – This is because only vertex-edge gradient vectors will contribute to the 1-unstable manifolds, and also new vertex-edge vectors can only be generated while canceling other vertex-edge pairs. Hence in PerSimpVF(), we can consider *only vertex-edge pairs* $(\sigma, \tau) \in P$ in order. Furthermore, it is not necessary to check whether the cancellation is valid or not – it will always be valid as long as the pairs are processed in increasing orders of persistence [3][1].

However, we can further simplify the algorithm as follows: First, we replace procedure PerSimpVF() by procedure PerSimpTree() as shown in Algorithm 2, which is much simpler both conceptually and implementation speaking. Note that there is *no explicit cancellation operation* any more.

The 1-dimensional simplicial complex $\mathcal{T}$ output by procedure PerSimpTree() may have multiple connected components $\mathcal{T} = \{T_1, \ldots, T_k\}$ – In fact, it is known that each T_i is a tree and $\mathcal{T}$ is a forest (see results from [2, 3] as summarized in Lemma 2 below). For each component T_i, we define its *sink*, denoted by $\text{si}(T_i)$, as the vertex $\mathsf{v}_i \in T_i$ with the lowest function $g_\rho = -\rho$ value. Lemma 2 also states that the sink of T_i would have been the *only critical simplex* among all simplices in T_i, if we had performed the δ-simplification as specified in procedure PerSimpVF(). Next, we replace procedure CollectOutputG() by procedure Treebased-OutputG() shown in Algorithm 2. We use MorseReconSimp() to denote our simplified version of Algorithm 1 (with PerSimpVF() replaced by PerSimpTree(), and CollectOutputG() replaced by Treebased-OutputG(). In summary, algorithm MorseReconSimp(K, ρ, δ) works by first computing all persistence pairs as before. It then collects all vertex-edge persistence pairs (v, e) with $\text{pers}(v, e) \leq \delta$. These edges along with the set of all vertices form a spanning forest $\mathcal{T}$. Then, for every edge $e = \langle u, v\rangle$ with $\text{pers}(e) > \delta$, it outputs the 1-unstable manifold of e, which is simply the union of e and the unique paths from u and v to the sink (root) of the tree containing them respectively. Its time complexity is stated below; note for the previous algorithm MorseRecon(), the cancellation step can take $\tilde{O}(n^2)$ time.

[1] We remark that though [3] states that the cancellation is not valid in higher dimension or non-manifold 2-complexes, all cancellations in PerSimpVF() are for vertex-edge pairs in a spanning tree which can be viewed as a 1-complex, and thus are always valid.

▶ **Theorem 1.** *The time complexity of our Algorithm* PerSimpVF*() is* $O(Pert(K)+n)$*, where* $PerT(K)$ *is the time to compute persistence pairings for* K*, and* n *is the total number of vertices and edges in* K*.*

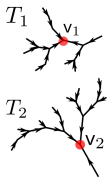
We remark that the $O(n)$ term is contributed by the step collecting all 1-unstable manifolds, which takes linear time if one avoids revisiting edges while tracing the paths.

Justification of the modified algorithm MorseReconSimp(). Let M_δ denote the resulting discrete gradient field after canceling all *vertex-edge* persistence pairs in $P(K)$ with persistence at most δ; that is, M_δ is the output of the procedure PerSimpVF() (although we only compute the relevant part of the discrete gradient vector field). Using observations from [2, 3], we show that the output $\mathcal{T}$ of procedure PerSimpTree() includes all information of M_δ. Furthermore, procedure Treebased-OutputG() computes the correct 1-unstable manifolds for all critical edges with persistence larger than δ. Indeed, observe that edges in Morse pairings from M_δ (for any $\delta \geq 0$) form a spanning forest of edges in K. Results of [3] imply that the output $\mathcal{T}$ constructed by our modified procedure corresponds exactly to this spanning forest:

▶ **Lemma 2.** *The following statements hold for the output* $\mathcal{T}$ *of procedure* PerSimpTree*() w.r.t any* $\delta \geq 0$*:*

- **(i)** $\mathcal{T}$ *is a spanning forest consisting of potentially multiple trees* $\{T_1, \ldots, T_k\}$*.*
- **(ii)** *For each tree* T_i*, its sink* v_i *is the only critical simplex in* M_δ*. The collection of* v_i*s corresponds exactly to those vertices whose persistence is bigger than* δ*.*
- **(iii)** *Any edge with* $\text{pers}(e) > \delta$ *remains critical in* M_δ *(and cannot be contained in* $\mathcal{T}$*).*

Note that, (ii) above implies that for each T_i, any discrete gradient path of M_δ in T_i terminates at its sink v_i. See the right figure for an illustration. Hence for any vertex $v \in T_i$, the path $\pi(v)$ computed in procedure Treebased-OutputG() is the unique discrete gradient path starting at v. This immediately leads to the following result:

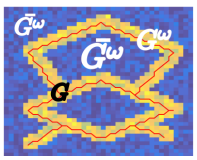


▶ **Corollary 3.** *For each critical edge* $e = \langle u, v\rangle$ *with* $\text{pers}(e) \geq \delta$*,* $\pi(u) \cup \pi(v) \cup \{e\}$ *as computed in procedure* Treebased-OutputG*() is the 1-unstable manifold of* e *in* M_δ*. Hence the output of our simplified algorithm* MorseReconSimp*() equals that of the original algorithm* MorseRecon*().*

4 Noise model

We first describe the noise model in the continuous setting where the domain is $\Omega = [0, 1]^d$. We then explain the setup in the discrete setting when the input is a triangulation K of Ω.

Given a connected "true graph" $\mathrm{G} \subset \Omega$, consider a ω-*neighborhood* $G^\omega \subseteq \Omega$, meaning that (i) $\mathrm{G} \subseteq G^\omega$, and (ii) for any $x \in G^\omega$, $d(x, \mathrm{G}) \leq \omega$ (i.e, G^ω is sandwiched between G and its ω-offset). Given G^ω, we use $\text{cl}(\overline{G^\omega})$ to denote the closure of its complement $\text{cl}(\overline{G^\omega}) = \text{cl}(\Omega \setminus G^\omega)$. See the right figure, showing G (red graph) with its ω-neighborhood G^ω (orange).



▶ **Definition 4.** A density function $\rho : \Omega \to \mathbb{R}$ is a (β, ν, ω)-*approximation of a connected graph* G if the following holds:

C-1 There is a ω-neighborhood G^ω of G such that G^ω deformation retracts to G.

C-2 $\rho(x) \in [\beta, \beta + \nu]$ for $x \in G^\omega$; and $\rho(x) \in [0, \nu]$ otherwise. Furthermore, $\beta > 2\nu$.

Intuitively, this noise model requires that the density ρ concentrates around the true graph G in the sense that the density is significantly higher inside G^ω than outside it; and the density fluctuation inside or outside G^ω is small compared to the density value in G^ω (condition C-2). Condition C-1 says that the neighborhood has the same topology of the hidden graph. Such a density field could for example be generated as follows: Imagine that there is an ideal density field $f_G : \Omega \to \mathbb{R}$ where $f_G(x) = \beta$ for $x \in G^\omega$ and 0 otherwise. There is a noisy perturbation $g : \Omega \to \mathbb{R}$ whose size is always bounded by $g(x) \in [0, \nu]$ for any $x \in \Omega$. The observed density field $\rho = f_G + g$ is an (β, ν, ω)-approximation of G.

In the discrete setting when we have a triangulation K of Ω, we define a ω-neighborhood G^ω to be a subcomplex of K, i.e, $G^\omega \subseteq K$, such that (i) G is contained in the underlying space of G^ω and (ii) for any vertex $v \in V(G^\omega)$, $d(v, \mathrm{G}) \leq \omega$. The outside-region $\mathrm{cl}(\overline{G^\omega}) \subseteq K$ is simply the smallest subcomplex of K that contains all simplices from $K \setminus G^\omega$ (i.e, all simplices **not** in G^ω and their faces). A PL-function $\rho : K \to \mathbb{R}$ (β, ν, ω)-approximation of G can be extended to this setting by requiring the underlying space of G^ω deformation retracts to G as in (C-1), and having those density conditions in (C-2) only at vertices of K.

We remark that the noise model is still limited – In particular, it does not allow significant non-uniform density distribution. However, this is the first time that theoretical guarantees are provided for a discrete Morse based reconstruction framework, despite that such a framework has been used for different applications before. We also give experiments and discussions in Appendix B of the full version [9] that the algorithm works beyond this noise model empirical, where thresholding type approaches do not work.

5 Theoretical guarantee

In this section, we prove results that are applicable to any dimension. Recall that M_δ is the discrete gradient field after the *δ-Morse cancellation process*, where we perform Morse-cancellation for all *vertex-edge* persistence pairs from $P(K)$. (While our algorithm does not maintain M_δ explicitly, we use it for theoretical analysis.) At this point, all positive edges (i.e, those paired with triangles or unpaired in $P(K)$) remain critical in M_δ. Some negative edges (i.e, those paired with vertices in $P(K)$) are also critical in M_δ – these are exactly the negative edges with persistence bigger than δ. Treebased-OutputG() only takes the 1-unstable manifolds of those critical edges (positive or negative) with persistence bigger than δ; so those positive edges whose persistence is $\leq \delta$ (if there is any) are ignored.

From now on, we use "*under our noise model*" to refer to (1) the input is a (β, ν, ω)-approximated density field w.r.t. G, and (2) $\delta \in [\nu, \beta - \nu)$. Let $\widehat{G}$ be the output of algorithm MorseReconSimp(K, ρ, δ). The proof of the following result is in the full version [9].

▶ **Proposition 5.** *Under our noise model, we have:*

- **(i)** *There is a single critical vertex left after* PerSimpVF*() which is in G^ω.*
- **(ii)** *Every critical edge considered by* Treebased-OutputG*() forms a persistence pair with a triangle.*
- **(iii)** *Every critical edge considered by* Treebased-OutputG*() is in G^ω.*

▶ **Theorem 6.** *Under our noise model, the output graph satisfies $\hat{G} \subseteq G^\omega$.*

Proof. Recall that the output graph $\widehat{G}$ consists of the union of 1-unstable manifolds of all the edges $e_1^*, \ldots, e_g^*$ with persistence larger than δ – By Propositions 5 (ii) and (iii), they are all positive (paired with triangles), and contained inside G^ω.

Take any $i \in [1, g]$ and consider $e_i^* = \langle u, v \rangle$. Without loss of generality, consider the gradient path starting from u: $\pi : u = u_1, e_1, u_2, e_2, \ldots, u_s, e_s, u_{s+1}$. By Lemma 2 and

Proposition 5, u_{s+1} must be a critical vertex (a sink) and is necessarily the global minimum v_0, which is also contained inside G^ω. We now argue that the entire path π (i.e, all simplices in it) is contained inside G^ω. In fact, we argue a stronger statement: First, we say that a gradient vector (v, e) is *crossing* if $v \in G^\omega$ and $e \notin G^\omega$ (i.e, $e \in \text{cl}(\overline{G^\omega})$) – Since v is an endpoint of e, this means that the other endpoint of e must lie in $K \setminus G^\omega$.

▶ **Claim 1.** *During the δ-Morse cancellation, no crossing gradient vector is ever produced.*

Proof. Suppose the lemma is not true: Then let (v, e) be the *first* crossing gradient vector ever produced during the δ-Morse cancellation process. Since we start with a trivial discrete gradient vector field, the creation of (v, e) can only be caused by reversing of some gradient path π' connecting two critical simplices v' and e' while we are performing Morse-cancellation for the persistence pair (v', e'). Obviously, $\text{pers}(v', e') \leq \delta$. On the other hand, due to our (β, ν, ω)-noise model and the choice of δ, it must be that either both $v', e' \in G^\omega$ or both $v', e' \in K \setminus G^\omega$ – as otherwise, the persistence of this pair will be larger than $\beta - \nu > \delta$.

Now consider this gradient path π' connecting v' and e' in the current discrete gradient vector field M'. Since the pair (v, e) becomes a gradient vector after the inversion of this path, it must be that (w, e) currently is a gradient vector where $e = \langle v, w \rangle$. Furthermore, since the path π' begins and ends with simplices either both in G^ω or both outside it, the path π' must contain a gradient vector (v'', e'') going in the opposite direction crossing inside/outside, that is, $v'' \in G^\omega$ and $e'' \notin G^\omega$. In other words, it must contain a crossing gradient vector. This however contradicts to our assumption that (v, e) would be the first crossing gradient vector. Hence the assumption is wrong and no crossing gradient vector can ever be created. ◀

As there is no crossing gradient vector during and after δ-Morse cancellation, it follows that π, which is one piece of the 1-unstable manifold of the critical edge e_i^*, has to be contained inside G^ω. The same argument works for the other piece of 1-unstable manifold of e_i^* (starting from the other endpoint of e_i^*). Since this is for any $i \in [1, g]$, the theorem holds. ◀

The previous theorem shows that $\widehat{G}$ is close to G in geometry. Next we will show that they are also close in topology.

▶ **Proposition 7.** *Under our noise model, $\widehat{G}$ is homotopy equivalent to G.*

Proof. We show that $\widehat{G}$ has the same first Betti number as that of G which implies the claim as any two graphs in $\mathbb{R}^d$ with the same first Betti number are homotopy equivalent.

The underlying space of ω-neighborhood G^ω of G deformation retracts to G by definition. Observe that, by our noise model, G^ω is a sublevel set in the filtration that determines the persistence pairs. This sublevel set being homotopy equivalent to G must contain exactly g positive edges where g is the first Betti number of G. Each of these positive edges pairs with a triangle in $\overline{G^\omega}$. Therefore, $\text{pers}(e, t) > \delta$ for each of the g positive edges in G^ω. By our earlier results, these are exactly the edges that will be considered by procedure Treebased-OutputG(). Our algorithm constructs $\widehat{G}$ by adding these g positive edges to the spanning tree each of which adds a new cycle. Thus, $\widehat{G}$ has first Betti number g. ◀

We have already proved that $\widehat{G}$ is contained in G^ω. This fact along with Proposition 7 can be used to argue that any deformation retraction taking (underlying space) G^ω to G also takes $\widehat{G}$ to a subset $G' \subseteq G$ where G' and G have the same first Betti number. In what follows, we use G^ω to denote also its underlying space.

▶ **Theorem 8.** *Let $F : G^\omega \times [0,1] \to G^\omega$ be any deformation retraction. Then, the restriction $F|_{\widehat{G}} : \widehat{G} \times [0,1] \to G^\omega$ is a homotopy from the embedding $\widehat{G}$ to $G' \subseteq G$ where G' is the minimal subset so that G and G' have the same first Betti number.*

Proof. The fact that $F|_{\widehat{G}}(\cdot, \ell)$ is continuous for any $\ell \in [0,1]$ is obvious from the continuity of F. Only thing that needs to be shown is that $F|_{\widehat{G}}(\widehat{G}, 1) = G'$. Suppose not. Then, $G'' = F|_{\widehat{G}}(\widehat{G}, 1)$ is a proper subset of G which has a first Betti number less than that of G. We observe that the cycle in $\widehat{G}$ created by a positive edge e along with the paths to the root of the spanning tree is also non-trivial in G^ω because this is a cycle created by adding the edge e during persistence filtration and the edge e is not killed in G^ω.Therefore, a cycle basis for $\widehat{G}$ is also a homology basis for G^ω. Since the map $F(\cdot, 1) : G^\omega \to G$ is a homotopy equivalence, it induces an isomorphism in the respective homology groups; in particular, a homology basis in G^ω is mapped to a homology basis in G. Therefore, the image $G'' = F|_{\widehat{G}}(\widehat{G}, 1)$ must have a basis of cardinality g if $\widehat{G}$ has first Betti number g. But, G'' cannot have a cycle basis of cardinality g if it is a proper subset of G' reaching a contradiction. ◀

6 Additional guarantee for 2D

For $\mathbb{R}^2$, we now show that G^ω actually deformation retracts to $\widehat{G}$, which is stronger than saying G and $\widehat{G}$ are homotopy equivalent. We are unable to prove this result for dimensions higher than 2, as our current proof needs that the edge-triangle persistence pairs can always be canceled (even though our algorithm does not depend on edge-triangle cancellations at all). It would be interesting, as a future work, to see whether a different approach can be developed to avoid this obstruction for the special case under our noise model. The main result of this section is as follows.

▶ **Theorem 9.** *Under our noise model, G^ω deformation retracts to G and $\hat{G}$.*

This main result follows from Proposition 10 and Theorem 11 below. To prove them, we will show that there exists a partition $\mathcal{R} := \{R_i\}$ of the set of triangles in K for which Theorem 11 holds. (This theorem is our main tool in establishing the deformation retract.) We first state the results below before giving their proofs. Let $B_i = \partial R_i$ where ∂ is the boundary operator operating on the 2-chain R_i. We also abuse the notations R_i and B_i to denote the geometric space that is the point-wise union of simplices in the respective chains. Let t_i be a triangle in R_i whose choice will be explained later. In the following, let H be the maximal set of edges in $\widehat{G}$ whose deletions do not eliminate a cycle (assume that a vertex is deleted only if all of its edges are deleted). Observe that H necessarily consists of negative edges forming "hairs" attached to the loops of $\widehat{G}$ and hence to $\cup_i B_i$ because of the following proposition.

▶ **Proposition 10.** *Under our noise model, $\widehat{G} = \cup B_i \bigcup H$.*

▶ **Theorem 11.** *Under our noise model, there exists a partition $\{R_i\}$ of triangles in K such that, there is a deformation retraction of $\cup_i (R_i \setminus t_i)$ to $\widehat{G}$ that comprises of two deformation retractions, one from $\cup_i (R_i \setminus t_i)$ to G^ω and another one from G^ω to $\cup_i B_i \bigcup H$ which is $\widehat{G}$.*

Now we describe the construction of a partition $\mathcal{R}$ of the triangles in K to prove Proposition 10 and Theorem 11. For technicality we assume that K is augmented to a triangulation of a sphere by putting a vertex v at infinity and joining it to the boundary of K with edges and triangles all of whom have function value ∞. Let $P(K)$ be the collection of persistence pairs of the form either (σ, τ) or (σ, ∞) generated from the lower-star filtration $F(K)$ as described before. Since K is 2-dimensional, each pair (σ, τ) is either a vertex-edge pair or an

edge-triangle pair. We order persistence pairs in $P(K)$ by their persistence, where ties are broken via the nested order in the filtration $F(K)$, and obtain:

$$P(K) = \{(\sigma_1, \tau_1), \ldots, (\sigma_n, \tau_n), (\alpha_1, \infty), \ldots, (\alpha_s, \infty)\}. \tag{1}$$

Starting with a trivial discrete gradient vector field M_0 where all simplices in K are critical, the algorithm PerSimpVF() performs Morse cancellations for the first $m \leq n$ persistence pairs $(\sigma_1, \tau_1), \ldots, (\sigma_m, \tau_m)$ in order where $\text{pers}(\sigma_m, \tau_m) \leq \delta$ but $(\sigma_{m+1}, \tau_{m+1}) > \delta$, Let M_i denote the gradient vector field after canceling (σ_i, τ_i). Recall that in the implementation of the algorithm we do not need to perform Morse cancellation for any edge-triangle pairs. However in this section, for the theoretical analysis, we will cancel edge-triangle pairs as well. Recall that a positive edge is one that creates a 1-cycle, namely, it is either paired with a triangle or unpaired; while a negative edge is one that destroys a 0-cycle (i.e, paired with a vertex).

Consider the ordered sequence of edge-triangle persistence pairs, $(e_1, t_1), \ldots, (e_n, t_n)$, which is a subsequence of the one in (1). Consider the sequence $t_1, \ldots, t_n$ of triangles in K ordered by the above sequence. Recall the standard persistence algorithm [11]. It implicitly associates a 2-chain with a triangle t when searching for the edge it is about to pair with. This 2-chain is non-empty if t is a destructor, and is empty otherwise. Let D_i denote this 2-chain associated with t_i for $i \in [1, n]$. Initially, the algorithm asserts $D_i = t_i$. At any stage, if D_i is not empty, the persistence algorithm identifies the edge e in the boundary ∂D_i that has been inserted into the filtration $F(K)$ most recently. If e has not been paired with anyone, the algorithm creates the persistence pair (e, t_i). Otherwise, if e has already been paired with a triangle, say $t_{i'}$, then D_i is updated with $D_i = D_i + D_{i'}$ and the search continues. Given an index $j \in [1, n]$, we define a modified set of chains C_i^j inductively as follows. For $j = 1$, $C_i^1 = t_i$. Assume that C_i^{j-1} has been already defined. To define C_i^j, similar to the persistence algorithm, check if the edge e_{j-1} is on the boundary ∂C_i^{j-1}. If so, define $C_i^j := C_i^{j-1} + C_{j-1}^{j-1}$ and $C_i^j := C_i^{j-1}$ otherwise. The following result is proved in [9].

▶ **Proposition 12.** *For $i \in [1, n]$, e_i is in ∂C_i^i. Furthermore, e_i is the most recent edge in ∂C_i^i according to the filtration order $F(K)$.*

Procedure PerSimpVF() also implicitly maintains a 2-chain R_i^* with each triangle t_i. Initially, $R_i^* = t_i$ as in the case of D_i. Then, inductively assume that R_i^* is the 2-chain implicitly associated with t_i when a persistence pair $(e_{i'}, t_{i'})$ is about to be considered by PerSimpVF() and the boundary ∂R_i^* contains $e_{i'}$. By reversing a gradient path between $t_{i'}$ and $e_{i'}$, it implicitly updates the 2-chain R_i^* as $R_i^* := R_i^* + R_{i'}^*$. We observe that R_i^* is identical with $C_i^{i'}$. Proposition 13 below establishes this fact along with some other inductive properties useful to prove Theorem 11. The proof can be found in the full version [9].

▶ **Proposition 13.** *Let (e_j, t_j) be the edge-triangle persistence pair* PerSimpVF() *is about to consider and let C_i^j be the 2-chains defined as above . Then, the following statements hold:*

- **(a)** *For each triangle t_i, $i = 1, \ldots, n$, in the persistence order, the 2-chain R_i^* satisfies the following conditions:(a.i) $R_i^* = C_i^j$, (a.ii) interpreting R_i^* as a set of triangles, one has that the sets R_i^*, $i = j, \ldots, n$, partition the set of all triangles in K.*
- **(b)** *There is a gradient path from t_i to all edges of the triangles in R_i^*, and (b.i) the path is unique if the edge is in the boundary ∂R_i^* for every $i = j, \ldots, n$; (b.ii) if there is more than one gradient path from t_i to an edge e, then e must be a negative edge.*

We are now ready to setup the regions R_is needed for Theorem 11 and Proposition 10. Suppose the first m edge-triangle pairs have persistence less than or equal to δ, the parameter

supplied to PerSimpVF(). Then, we set $R_i = R_i^*$ as in Proposition 13 for $i \geq m+1, \ldots, n$. The proof for Proposition 10 is in the full version [9].

Finally, similar to the vertex-edge gradient vectors, we say that a gradient vector (e, t) is *crossing* if $e \in G^\omega$ and $t \notin G^\omega$. The following claim can be proved similarly as Claim 1.

▶ **Claim 2.** *During the δ-Morse cancellation of edge-triangle pairs, no crossing gradient vector is ever produced.*

Proof of Theorem 11. Set $\hat{R}_i = R_i \setminus t_i$. Let $\mathcal{T}$ be the spanning tree formed by all negative edges and their vertices. Let L_i be the set of edges in R_i that has more than one gradient path from t_i to them; $L_i \subset \mathcal{T}$ by Proposition 13 (b.ii). First, we want to establish a deformation retraction from $\cup(\hat{R}_i \setminus t_i)$ to G^ω. To do this, for $k = 0, 1, \ldots, s$, we will define $\hat{R}_i^k$ inductively where $\hat{R}_i^{k-1}$ deformation retracts to $\hat{R}_i^k$ and $\hat{R}_i^s \subseteq G^\omega \cup L_i$. Let $\hat{R}_i^0 = \hat{R}_i$. For $k = 1, \ldots, s$, consider a *positive* edge e in $\hat{R}_i^{k-1}$ where (a) e is not in G^ω and (b) there is a unique gradient path in R_i from t_i to e that passes through triangles all of which are in $R_i \setminus \hat{R}_i^{k-1}$. If such an edge e exists, then e is necessarily incident to a single triangle, say t, in $\hat{R}_i^{k-1}$. We collapse the pair (e, t), which is necessarily an edge-triangle gradient vector pair because e is positive. We take $\hat{R}_i^k$ to be $\hat{R}_i^{k-1} \setminus \{e, t\}$. If no such e exists, then either (A) there is no positive edge in $\hat{R}_i^{k-1} \setminus G^\omega$ any more; or (B) for each positive edge $e' \in \hat{R}_i^{k-1} \setminus G^\omega$, (B-1) there is a unique gradient path from t_i to e' but this path passes through some triangle in $\hat{R}_i^{k-1}$; or (B-2) there are two gradient paths from t_i to e'.

If there is no positive edge in $\hat{R}_i^{k-1} \setminus G^\omega$ any more, then $\hat{R}_i^{k-1} \subseteq G^\omega \cup L_i$, as otherwise, there will be at least some triangle from $\hat{R}_i^{k-1} \setminus G^\omega \cup L_i$ with at least one boundary edge of it being positive. The induction then terminates; we set $s = k - 1$ and reach our goal.

We now show that case (B-1) is not possible. Suppose it happens, that is, $e' \in \hat{R}_i^{k-1} \setminus L_i$ is an edge not in G^ω for which the unique gradient path from t_i passes through triangles in $\hat{R}_i^{k-1}$. Let e'' be the first edge in this path that is in $\hat{R}_i^{k-1} \setminus L_i$. Then, if $e'' \notin G^\omega$, it qualifies for the conditions (a) and (b) required for e reaching a contradiction. So, assume $e'' \in G^\omega$. But, in that case, we have a gradient path that goes into G^ω and then comes out to reach $e' \notin G^\omega$. There has to be a gradient pair in this path where the edge is in G^ω and the triangle is not in G^ω. This contradicts Claim 2. Thus, case (B-1) is not possible. Now consider (B-2): e' must be negative by Proposition 13 (b.ii). So, it is not possible either.

To summarize, the induction terminates in case (A), at which time we would have that $\hat{R}_i^s \subseteq G^\omega \cup L_i$. Furthermore, this process also establishes a deformation retraction from $\hat{R}_i$ to $\hat{R}_i^s$ realized by successive collapses of edge-triangle pairs. Furthermore, by construction, each collapsed pair (e, t) must be from $\mathrm{cl}(\overline{G^\omega})$, hence $\cup_i \hat{R}_i^s$ contains all simplices in G^ω. Combined with that $\hat{R}_i^s \subseteq G^\omega \cup L_i$, we have that $\cup_i \hat{R}_i^s = G^\omega \cup L$, where $L = \cup_i L_i$ is a subset of the spanning tree $\mathcal{T}$. The edges in L being part of a spanning tree cannot form a cycle and thus can be retracted along the tree to G^ω, which gives rise to a deformation retraction from $\cup_i(R_i \setminus t_i)$ to $G^\omega \cup L$ and then to G^ω, establishing the first part of Theorem 11.

We now show that $(\cup_i \hat{R}_i^s) \bigcap G^\omega = G^\omega$ deformation retracts to $\cup B_i \bigcup H$. Let $\hat{L}_i$ be the edges in $\hat{R}_i^s \cap G^\omega$ with more than one gradient path from t_i to them. These edges are negative by Proposition 13 (b.ii). Replacing $\hat{R}_i$ with $\hat{R}_i^s \cap G^\omega$ and edges in $B_i \cup \hat{L}_i$ playing the role of edges in $G^\omega \cup L_i$ in the above induction, we can obtain that $\hat{R}_i^s \cap G^\omega$ deformation retracts to $B_i \cup \hat{L}_i$. Observe that now instead of Claim 2, we use the fact that no edge-triangle gradient path crosses B_i that consists of only negative and critical edges. To this end, we also observe that $\cup \hat{L}_i = \mathcal{T} \cap G^\omega$ where $\mathcal{T}$ is the spanning tree formed by all negative edges, as we only collapse edge-triangle pairs that are gradient pairs (hence the participating edges are always positive). This implies that $H \subset \cup \hat{L}_i$. Again, edges in $\hat{L}_i$ (being part of a spanning tree) can

be retracted along the spanning tree till one reaches B_i or edges in H. Performing this for each i, we thus obtain a deformation retraction from $(\cup_i \hat{R}_i^s) \bigcap G^\omega = G^\omega$ to $\cup B_i \bigcup \cup \hat{L}_i$and further to $\cup B_i \bigcup H = \widehat{G}$. This finishes the proof of Theorem 11. ◀

In the full version of this paper [9], we also provide some experiments demonstrating the efficiency of the simplified algorithm, as well as discussion on thresholding strategies.

References

1 M. Aanjaneya, F. Chazal, D. Chen, M. Glisse, L. Guibas, and D. Morozov. Metric graph reconstruction from noisy data. *International Journal of Computational Geometry & Applications*, 22(04):305–325, 2012.

2 D. Attali, M. Glisse, S. Hornus, F. Lazarus, and D. Morozov. Persistence-sensitive simplification of functions on surfaces in linear time. *Presented at TOPOINVIS*, 9:23–24, 2009.

3 U. Bauer, C. Lange, and M. Wardetzky. Optimal topological simplification of discrete functions on surfaces. *Discr. Comput. Geom.*, 47(2):347–377, 2012.

4 S. Biasotti, D. Giorgi, M. Spagnuolo, and B. Falcidieno. Reeb graphs for shape analysis and applications. *Theoretical Computer Science*, 392(1-3):5–22, 2008.

5 F. Chazal, R. Huang, and J. Sun. Gromov–hausdorff approximation of filamentary structures using reeb-type graphs. *Discr. Comput. Geom.*, 53(3):621–649, 2015.

6 O. Delgado-Friedrichs, V. Robins, and A. Sheppard. Skeletonization and partitioning of digital images using discrete morse theory. *IEEE Trans. Pattern Anal. Machine Intelligence*, 37(3):654–666, March 2015.

7 T. Dey and J. Sun. Defining and computing curve-skeletons with medial geodesic function. In *Sympos. Geom. Proc.*, volume 6, pages 143–152, 2006.

8 T. Dey, J. Wang, and Y. Wang. Improved road network reconstruction using discrete morse theory. In *Proc. 25th ACM SIGSPATIAL*. ACM, 2017.

9 T. Dey, J. Wang, and Y. Wang. Graph reconstruction by discrete morse theory. *arXiv preprint arXiv:1803.05093*, 2018.

10 H. Edelsbrunner and J. Harer. *Computational Topology - an Introduction*. American Mathematical Soc., 2010. URL: `http://www.ams.org/bookstore-getitem/item=MBK-69`.

11 H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discr. Comput. Geom.*, 28:511–533, 2002.

12 The ENZO project. `http://enzo-project.org`.

13 The Center for Integrative Biomedical Computing (CIBC). Micro-CT Dataset Archive. `https://www.sci.utah.edu/cibc-software/cibc-datasets.html`.

14 R. Forman. Morse theory for cell complexes. *Advances in mathematics*, 134(1):90–145, 1998.

15 X. Ge, I. I Safa, M. Belkin, and Y. Wang. Data skeletonization via reeb graphs. In *Advances in Neural Info. Proc. Sys.*, pages 837–845, 2011.

16 A. Gyulassy, M. Duchaineau, V. Natarajan, V. Pascucci, E. Bringa, A. Higginbotham, and B. Hamann. Topologically clean distance fields. *IEEE Trans. Visualization Computer Graphics*, 13(6):1432–1439, Nov 2007.

17 T. Hastie and W. Stuetzle. Principal curves. *Journal of the American Statistical Association*, 84(406):502–516, 1989.

18 B. Kégl and A. Krzyzak. Piecewise linear skeletonization using principal curves. *IEEE Trans. Pattern Anal. Machine Intelligence*, 24(1):59–74, 2002.

19 L. Liu, E. W Chambers, D. Letscher, and T. Ju. Extended grassfire transform on medial axes of 2d shapes. *Computer-Aided Design*, 43(11):1496–1505, 2011.

20 J. Milnor. *Morse Theory*. Princeton Univ. Press, New Jersey, 1963.

21 M. Natali, S. Biasotti, G. Patanè, and B. Falcidieno. Graph-based representations of point clouds. *Graphical Models*, 73(5):151–164, 2011.

22 U. Ozertem and D. Erdogmus. Locally defined principal curves and surfaces. *Journal of Machine learning research*, 12(Apr):1249–1286, 2011.

23 V. Robins, P. J. Wood, and A. P. Sheppard. Theory and algorithms for constructing discrete morse complexes from grayscale digital images. *IEEE Trans. Pattern Anal. Machine Intelligence*, 33(8):1646–1658, Aug 2011.

24 T. Sousbie. The persistent cosmic web and its filamentary structure–i. theory and implementation. *Monthly Notices of the Royal Astronomical Society*, 414(1):350–383, 2011.

25 S. Wang, Y. Wang, and Y. Li. Efficient map reconstruction and augmentation via topological methods. In *Proc. 23rd ACM SIGSPATIAL*, page 25. ACM, 2015.

26 Y. Yan, K. Sykes, E. Chambers, D. Letscher, and T. Ju. Erosion thickness on medial axes of 3d shapes. *ACM Trans. on Graphics*, 35(4):38, 2016.

Computing Bottleneck Distance for 2-D Interval Decomposable Modules

Tamal K. Dey
Department of Computer Science and Engineering, The Ohio State University, Columbus, Ohio, USA
http://web.cse.ohio-state.edu/~dey.8/
tamaldey@cse.ohio-state.edu

Cheng Xin
Department of Computer Science and Engineering, The Ohio State University, Columbus, Ohio, USA
http://web.cse.ohio-state.edu/~xin.108/
xin.108@buckeyemail.osu.edu

Abstract

Computation of the interleaving distance between persistence modules is a central task in topological data analysis. For 1-D persistence modules, thanks to the isometry theorem, this can be done by computing the bottleneck distance with known efficient algorithms. The question is open for most n-D persistence modules, $n > 1$, because of the well recognized complications of the indecomposables. Here, we consider a reasonably complicated class called 2-*D interval decomposable* modules whose indecomposables may have a description of non-constant complexity. We present a polynomial time algorithm to compute the bottleneck distance for these modules from indecomposables, which bounds the interleaving distance from above, and give another algorithm to compute a new distance called *dimension distance* that bounds it from below.

2012 ACM Subject Classification Mathematics of computing → Topology, Theory of computation → Computational geometry

Keywords and phrases Persistence modules, bottleneck distance, interleaving distance

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.32

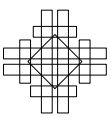
Related Version A full version of the paper is available at http://arxiv.org/abs/1803.02869

Funding This research is supported by NSF grants CCF-174061, CCF-1526513, and DMS-1547357

Acknowledgements We thank the anonymous referees for insightful comments on a previous version.

1 Introduction

Persistence modules have become an important object of study in topological data analysis in that they serve as an intermediate between the raw input data and the output summarization with persistence diagrams. The classical persistence theory [19] for $\mathbb{R}$-valued functions produces one dimensional (1-D) persistence modules, which is a sequence of vector spaces (homology groups with a field coefficient) with linear maps over $\mathbb{R}$ seen as a poset. It is known that [16, 26], this sequence can be decomposed uniquely into a set of intervals called *bars* which is also represented as points in $\mathbb{R}^2$ called the persistence diagrams [15]. The space of these diagrams can be equipped with a metric d_B called the *bottleneck distance*. Cohen-Steiner et al. [15] showed that d_B is bounded from above by the input function

34th International Symposium on Computational Geometry (SoCG 2018).
Editors: Bettina Speckmann and Csaba D. Tóth; Article No. 32; pp. 32:1–32:15


Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

perturbation measured in infinity norm. Chazal et al. [12] generalized the result by showing that the bottleneck distance is bounded from above by a distance d_I called the *interleaving distance* between two persistence modules; see also [6, 8, 17] for further generalizations. Lesnick [22] (see also [2, 13]) established the isometry theorem which showed that indeed $d_I = d_B$. Consequently, d_I for 1-D persistence modules can be computed exactly by efficient algorithms known for computing d_B; see e.g. [19, 20]. The status however is not so well settled for multidimensional (n-D) persistence modules [9] arising from $\mathbb{R}^n$-valued functions.

Extending the concept from 1-D modules, Lesnick defined the interleaving distance for multidimensional (n-D) persistence modules, and proved its stability and universality [22]. The definition of the bottleneck distance, however, is not readily extensible mainly because the bars for finitely presented n-D modules called *indecomposables* are far more complicated though are guaranteed to be essentially unique by Krull-Schmidt theorem [1]. Nonetheless, one can define d_B as the supremum of the pairwise interleaving distances between indecomposables, which in some sense generalizes the concept in 1-D due to the isometry theorem. Then, straightforwardly, $d_I \leq d_B$ as observed in [7], but the converse is not necessarily true. For some special cases, results in the converse direction have started to appear. Botnan and Lesnick [7] proved that, in 2-D, $d_B \leq \frac{5}{2} d_I$ for what they called block decomposable modules. Bjerkevic [4] improved this result to $d_B \leq d_I$. Furthermore, he extended it by proving that $d_B \leq (2n-1)d_I$ for rectangle decomposable n-D modules and $d_B \leq (n-1)d_I$ for free n-D modules. He gave an example for exactness of this bound when $n = 2$.

Unlike 1-D modules, the question of estimating d_I for n-D modules through efficient algorithms is largely open [5]. Multi-dimensional matching distance introduced in [10] provides a lower bound to interleaving distance [21] and can be approximated within any error threshold by algorithms proposed in [3, 11]. But, it cannot provide an upper bound like d_B. For free, block, rectangle, and triangular decomposable modules, one can compute d_B by computing pairwise interleaving distances between indecomposables in constant time because they have a description of constant complexity. Due to the results mentioned earlier, d_I can be estimated within a constant or dimension-dependent factors by computing d_B for these modules. It is not obvious how to do the same for the larger class of interval decomposable modules mentioned in the literature [4, 7] where indecomposables may not have constant complexity. These are modules whose indecomposables are bounded by "stair-cases". Our main contribution is a polynomial time algorithm that, given indecomposables, computes d_B exactly for 2-D interval decomposable modules. The algorithm draws upon various geometric and algebraic analysis of the interval decomposable modules that may be of independent interest. It is known that no lower bound in terms of d_B for d_I may exist for these modules [7]. To this end, we complement our result by proposing a distance d_0 called *dimension distance* that is efficiently computable and satisfies the condition $d_0 \leq d_I$.

All missing proofs of this article appear in the full version [18].

2 Persistence modules

Our goal is to compute the bottleneck distance between two 2-D interval decomposable modules. The bottleneck distance, originally defined for 1-D persistence modules [15] (also see [2]), and later extended to multi-dimensional persistence modules [7] is known to bound the interleaving distance between two persistence modules from above.

Let $\mathbb{F}$ be a field, **Vec** be the category of vector spaces over $\mathbb{F}$, and **vec** be the subcategory of finite dimensional vector spaces. In what follows, for simplicity, we assume $\mathbb{F} = \mathbb{Z}/2\mathbb{Z}$.

▶ **Definition 1** (Persistence module)**.** Let $\mathbb{P}$ be a poset category. A $\mathbb{P}$-indexed persistence module is a functor $M : \mathbb{P} \to \mathbf{Vec}$. If M takes values in **vec**, we say M is pointwise finite dimensional (p.f.d). The $\mathbb{P}$-indexed persistence modules themselves form another category where the natural transformations between functors constitute the morphisms.

Here we consider the poset category to be $\mathbb{R}^n$ with the standard partial order and all modules to be p.f.d. We call $\mathbb{R}^n$-indexed persistence modules as n-dimensional persistence modules, n-D modules in short. The category of n-D modules is denoted as $\mathbb{R}^n$-**mod**. For an n-D module $M \in \mathbb{R}^n$-**mod**, we use notation $M_x := M(x)$ and $\rho^M_{x\to y} := M(x \le y)$.

▶ **Definition 2** (Shift)**.** For any $\delta \in \mathbb{R}$, we denote $\vec{\delta} = \delta \cdot \sum e_i$, where $\{e_i\}_{i=1}^n$ is the standard basis of $\mathbb{R}^n$. We define a shift functor $(\cdot)_{\to\delta} : \mathbb{R}^n$-**mod** $\to \mathbb{R}^n$-**mod** where $M_{\to\delta} := (\cdot)_{\to\delta}(M)$ is given by $M_{\to\delta}(x) = M(x + \vec{\delta})$ and $M_{\to\delta}(x \le y) = M(x + \vec{\delta} \le y + \vec{\delta})$. In words, $M_{\to\delta}$ is the module M shifted diagonally by $\vec{\delta}$.

The following definition of interleaving taken from [24] adapts the original definition designed for 1-D modules in [13] to n-D modules.

▶ **Definition 3** (Interleaving)**.** For two persistence modules M and N, and $\delta \ge 0$, a δ-interleaving between M and N are two families of linear maps $\{\phi_x : M_x \to N_{x+\vec{\delta}}\}_{x\in\mathbb{R}^n}$ and $\{\psi_x : N_x \to M_{x+\vec{\delta}}\}_{x\in\mathbb{R}^n}$ satisfying the following two conditions (see full version [18] for the details.)

- $\forall x \in \mathbb{R}^n, \rho^M_{x\to x+2\vec{\delta}} = \psi_{x+\vec{\delta}} \circ \phi_x$ and $\rho^N_{x\to x+2\vec{\delta}} = \phi_{x+\vec{\delta}} \circ \psi_x$
- $\forall x \le y \in \mathbb{R}^n, \phi_y \circ \rho^M_{x\to y} = \rho^N_{x\to y} \circ \phi_x$ and $\psi_y \circ \rho^N_{x\to y} = \rho^M_{x\to y} \circ \psi_x$ symmetrically

If such a δ-interleaving exists, we say M and N are δ-interleaved. We call the first condition *triangular commutativity* and the second condition *square commutativity*.

▶ **Definition 4** (Interleaving distance)**.** Define the interleaving distance between modules M and N as $d_I(M, N) = \inf_\delta\{M \text{ and } N \text{ are } \delta\text{-}interleaved\}$. We say M and N are ∞-interleaved if they are not δ-interleaved for any $\delta \in \mathbb{R}^+$, and assign $d_I(M, N) = \infty$.

▶ **Definition 5** (Matching)**.** A matching $\mu : A \nrightarrow B$ between two multisets A and B is a partial bijection, that is, $\mu : A' \to B'$ for some $A' \subseteq A$ and $B' \subseteq B$. We say $\operatorname{im}\mu = B', \operatorname{coim}\mu = A'$.

For the next definition [7], we call a module δ-*trivial* if $\rho^M_{x\to x+\vec{\delta}} = 0$ for all $x \in \mathbb{R}^n$.

▶ **Definition 6** (Bottleneck distance)**.** Let $M \cong \bigoplus_{i=1}^m M_i$ and $N \cong \bigoplus_{j=1}^n N_j$ be two persistence modules, where M_i and N_j are indecomposable submodules of M and N respectively. Let $I = \{1, \cdots, m\}$ and $J = \{1, \cdots, n\}$. We say M and N are δ-matched for $\delta \ge 0$ if there exists a matching $\mu : I \nrightarrow J$ so that, (i) $i \in I \setminus \operatorname{coim}\mu \implies M_i$ is 2δ-trivial, (ii) $j \in J \setminus \operatorname{im}\mu \implies N_j$ is 2δ-trivial, and (iii) $i \in \operatorname{coim}\mu \implies M_i$ *and* $N_{\mu(i)}$ *are* δ-interleaved.

The bottleneck distance is defined as

$$d_B(M, N) = \inf\{\delta \mid M \text{ and } N \text{ are } \delta\text{-matched}\}.$$

The following fact observed in [7] is straightforward from the definition.

▶ **Fact 7.** $d_I \le d_B$.

2.1 Interval decomposable modules

Persistence modules whose indecomposables are interval modules (Definition 9) are called *interval decomposable modules*, see for example [7]. To account for the boundaries of free modules, we enrich the poset $\mathbb{R}^n$ by adding points at $\pm\infty$ and consider the poset $\bar{\mathbb{R}}^n = \bar{\mathbb{R}} \times \ldots \times \bar{\mathbb{R}}$ where $\bar{\mathbb{R}} = \mathbb{R} \cup \{\pm\infty\}$ with the usual additional rule $a \pm \infty = \pm\infty$.

▶ **Definition 8.** An interval is a subset $\emptyset \neq I \subset \bar{\mathbb{R}}^n$ that satisfies the following:

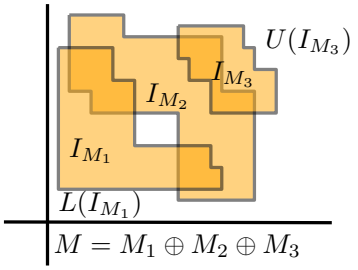
1. If $p, q \in I$ and $p \leq r \leq q$, then $r \in I$;
2. If $p, q \in I$, then there exists a sequence $(p_1, p_2, ..., p_{2m}) \in I$ for some $m \in \mathbb{N}$ such that $p \leq p_1 \geq p_2 \leq p_3 \geq ... \geq p_{2m} \leq q$. We call the sequence $(p = p_0, p_1, p_2, ..., p_{2m}, p_{2m+1} = q)$ a path from p to q (in I).

In what follows, we fix the dimension $n = 2$. Let $\bar{I}$ denote the closure of an interval I in the standard topology of $\bar{\mathbb{R}}^2$. The lower and upper boundaries of I are defined as

$$L(I) = \{x = (x_1, x_2) \in \bar{I} \mid \forall y = (y_1, y_2) \text{ with } y_1 < x_1 \text{ and } y_2 < x_2 \implies y \notin I\}$$
$$U(I) = \{x = (x_1, x_2) \in \bar{I} \mid \forall y = (y_1, y_2) \text{ with } y_1 > x_1 \text{ and } y_2 > x_2 \implies y \notin I\}.$$

See the figure below. Let $B(I) = L(I) \cup U(I)$.

We say an interval I is discretely presented if its boundary consists of a finite set of horizontal and vertical line segments called edges, with end points called vertices, which satisfy the following conditions: (i) every vertex is incident to either a single edge or to a horizontal and a vertical edge, (ii) no vertex appears in the interior of an edge. We denote the set of edges and vertices with $E(I)$ and $V(I)$ respectively.



According to this definition, $\bar{\mathbb{R}}^2$ is an interval with boundary $B(\bar{\mathbb{R}}^2)$ that consists of all the points with at least one coordinate ∞. The vertex set $V(\bar{\mathbb{R}}^2)$ consists of four corners of the infinitely large square $\bar{\mathbb{R}}^2$ with coordinates $(\pm\infty, \pm\infty)$.

▶ **Definition 9** (Interval module)**.** A 2-D **interval persistence module**, or *interval module* in short, is a persistence module M that satisfies the following condition: for some interval $I_M \subseteq \bar{\mathbb{R}}^2$, called the interval of M,

$$M_x = \begin{cases} \mathbb{k} & \text{if } x \in I_M \\ 0 & otherwise \end{cases} \qquad \rho^M_{x \to y} = \begin{cases} \mathbb{1}_{\mathbb{k}} & \text{if } x, y \in I_M \\ 0 & otherwise \end{cases}$$

It is known that an interval module is indecomposable [22].

▶ **Definition 10** (Interval decomposable module)**.** A 2-D *interval decomposable module* is a persistence module that can be decomposed into interval modules. We say a 2-D interval decomposable module is finitely presented if it can be decomposed into finitely many interval modules whose intervals are discretely presented.

3 Algorithm to compute d_B

Given the intervals of the indecomposables (interval modules) as input, an approach based on bipartite-graph matching is well known for computing the bottleneck distance $d_B(M, N)$ between two 1-D persistence modules M and N [19]. This approach constructs a bi-partite

graph G out of the intervals of M and N and their pairwise interleaving distances including the distances to zero modules. If these distance computations take $O(C)$ time in total, the algorithm for computing d_B takes time $O(m^{\frac{5}{2}} \log m + C)$ if M and N together have m indecomposables altogether. Given indecomposables (say computed by Meat-Axe [23]), this approach is readily extensible to the n-D modules if one can compute the interleaving distance between any pair of indecomposables including the zero modules. To this end, we present an algorithm to compute the interleaving distance between two interval modules M_i and N_j with t_i and t_j vertices respectively on their intervals in $O((t_i + t_j)\log(t_i + t_j))$ time. This gives a total time of $O(m^{\frac{5}{2}} \log m + \sum_{i,j}(t_i + t_j)\log(t_i + t_j)) = O(m^{\frac{5}{2}} \log m + t^2 \log t)$ where t is the number of vertices over all input intervals.

Now we focus on computing the interleaving distance between two given intervals. Given two intervals I_M and I_N with t vertices, this algorithm searches a value δ so that there exists two families of linear maps from M to $N_{\to\delta}$ and from N to $M_{\to\delta}$ respectively which satisfy both triangular and square commutativity. This search is done with a binary probing. For a chosen δ from a candidate set of $O(t)$ values, the algorithm determines the direction of the search by checking two conditions called *trivializability* and *validity* on the intersections of modules M and N.

▶ **Definition 11** (Intersection module)**.** For two interval modules M and N with intervals I_M and I_N respectively let $I_Q = I_M \cap I_N$, which is a disjoint union of intervals, $\coprod I_{Q_i}$. The intersection module Q of M and N is $Q = \bigoplus Q_i$, where Q_i is the interval module with interval I_{Q_i}. That is,

$$Q_x = \begin{cases} \urcorner & \text{if } x \in I_M \cap I_N \\ 0 & otherwise \end{cases} \quad \text{and for } x \le y,\ \rho^Q_{x\to y} = \begin{cases} \nVdash & \text{if } x, y \in I_M \cap I_N \\ 0 & otherwise \end{cases}$$

From the definition we can see that the support of Q, $supp(Q)$, is $I_M \cap I_N$. We call each Q_i an intersection component of M and N. Write $I := I_{Q_i}$ and consider $\phi : M \to N$ to be any morphism in the following proposition which says that ϕ is constant on I.

▶ **Proposition 12.** $\phi|_I \equiv a \cdot \nVdash$ for some $a \in \urcorner = \mathbb{Z}/2$.

Proof.

$$\begin{array}{ccc} M_{p_i} & \xrightarrow{\nVdash} & M_{p_{i+1}} \\ {\scriptstyle \phi_{p_i}}\downarrow & & \downarrow{\scriptstyle \phi_{p_{i+1}}} \\ N_{p_i} & \xrightarrow[\nVdash]{} & N_{p_{i+1}} \end{array} \qquad \begin{array}{ccc} M_{p_i} & \xleftarrow{\nVdash} & M_{p_{i+1}} \\ {\scriptstyle \phi_{p_i}}\downarrow & & \downarrow{\scriptstyle \phi_{p_{i+1}}} \\ N_{p_i} & \xleftarrow[\nVdash]{} & N_{p_{i+1}} \end{array}$$

For any $x, y \in I$, consider a path $(x = p_0, p_1, p_2, ..., p_{2m}, p_{2m+1} = y)$ in I from x *to* y and the commutative diagrams above for $p_i \le p_{i+1}$ (left) and $p_i \ge p_{i+1}$(right) respectively. Observe that $\phi_{p_i} = \phi_{p_{i+1}}$ in both cases due to the commutativity. Inducting on i, we get that $\phi(x) = \phi(y)$. ◀

▶ **Definition 13** (Valid intersection)**.** An intersection component Q_i is (M, N)-*valid* if for each $x \in I_{Q_i}$ the following two conditions hold (see figure below):

(i) $y \le x$ and $y \in I_M \implies y \in I_N$, and (ii) $z \ge x$ and $z \in I_N \implies z \in I_M$

▶ **Proposition 14.** Let $\{Q_i\}$ be a set of intersection components of M and N with intervals $\{I_{Q_i}\}$. Let $\{\phi_x\} : M \to N$ be the family of linear maps defined as $\phi_x = \nVdash$ for all $x \in I_{Q_i}$ and $\phi_x = 0$ otherwise. Then ϕ is a morphism if and only if every Q_i is (M, N)-*valid*.

For proof see the full version [18].

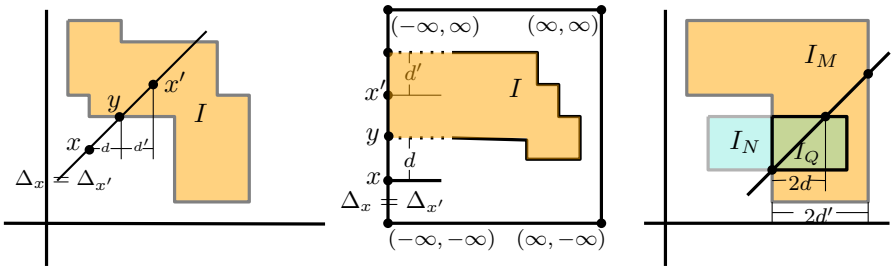
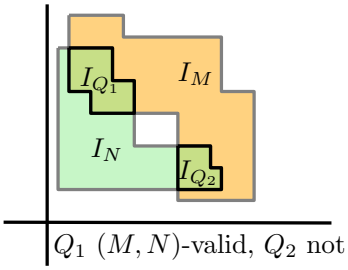


Figure 1 $d = \mathrm{dl}(x, I)$, $y = \pi_I(x)$, $d' = \mathrm{dl}(x', L(I))$ (left); $d = \mathrm{dl}(x, I)$ and $d' = \mathrm{dl}(x', U(I))$ are defined on the left edge of $B(\bar{\mathbb{R}}^2)$ (middle); Q is $d'_{(M,N)}$- and $d_{(N,M)}$-trivializable (right).

We focus on the interval modules with discretely presented intervals (figure on right). They belong to finitely presented persistence modules studied previously in [22]. For an interval module M, let $\overline{M}$ be the interval module defined on the closure $\overline{I_M}$. To avoid complication in this exposition, we assume that the upper and lower boundaries of every interval module meet exactly at two points. We also assume that every interval module has closed intervals which is justified by the following proposition (proof in the full version [18]).



▶ **Proposition 15.** $d_I(M, N) = d_I(\overline{M}, \overline{N})$.

From the definition of boundaries of intervals, the following proposition is immediate.

▶ **Proposition 16.** Given an interval I and any point $x = (x_1, x_2) \in I \setminus (I \cap B(\bar{\mathbb{R}}^2))$, we have $x \in L(I) \iff \forall \epsilon > 0, x - \vec{\epsilon} \notin I$. Similarly, we have $x \in U(I) \iff \forall \epsilon > 0, x + \vec{\epsilon} \notin I$.

▶ **Definition 17** (Diagonal projection and distance)**.** Let I be an interval and $x \in \bar{\mathbb{R}}^2$. For $x \in \mathbb{R}^2 \subseteq \bar{\mathbb{R}}^2$, let Δ_x denote the line called *diagonal* with slope 1 that passes through x. We define (see Figure 1)

$$\mathrm{dl}(x, I) = \begin{cases} \min_{y \in \Delta_x \cap I} \{d_\infty(x, y) := |x - y|_\infty\} \text{ if } \Delta_x \cap I \neq \emptyset \\ +\infty \text{ otherwise.} \end{cases}$$

In case $\Delta_x \cap I \neq \emptyset$, define $\pi_I(x)$, called the projection point of x on I, to be the point $y \in \Delta_x \cap I$ where $\mathrm{dl}(x, I) = d_\infty(x, y)$. For $x \in B(\bar{\mathbb{R}}^2) \setminus V(\bar{\mathbb{R}}^2)$, Δ_x is defined to be the edge in $E(\bar{\mathbb{R}}^2)$ containing x. Define $\mathrm{dl}(x, I)$ and $\pi_I(x)$ accordingly. For $x \in V(\bar{\mathbb{R}}^2)$, we set $\pi_I(x) = x$ if and only if $x \in I$. Then, $\mathrm{dl}(x, I) = 0$ if $x \in I$ and $\mathrm{dl}(x, I) = +\infty$ otherwise.

Notice that upper and lower boundaries of an interval are also intervals by definition. With this understanding, following properties of dl are obvious from the above definition.

▶ **Fact 18.**

(i) For any $x \in I_M$,

$$\mathrm{dl}(x, U(I_M)) = \sup_{\delta \in \bar{\mathbb{R}}} \{x + \vec{\delta} \in I_M\} \text{ and } \mathrm{dl}(x, L(I_M)) = \sup_{\delta \in \bar{\mathbb{R}}} \{x - \vec{\delta} \in I_M\}.$$

(ii) Let $L = L(I_M)$ or $U(I_M)$ and let x, x' be two points such that $\pi_L(x), \pi_L(x')$ both exist. If x and x' are on some same horizontal, vertical, or diagonal line, then $|\mathrm{dl}(x, L) - \mathrm{dl}(x', L)| \leq d_\infty(x, x')$.

Set $VL(I) := V(I) \cap L(I)$, $EL(I) := E(I) \cap L(I)$, $VU(I) := V(I) \cap U(I)$, and $EU(I) := E(I) \cap U(I)$. Following proposition is proved in the full version [18].

▶ **Proposition 19.** For an intersection component Q of M and N with interval I, the following conditions are equivalent:
(1) Q is (M, N)-*valid*.
(2) $L(I) \subseteq L(I_M)$ and $U(I) \subseteq U(I_N)$.
(3) $VL(I) \subseteq L(I_M)$ and $VU(I) \subseteq U(I_N)$.

▶ **Definition 20** (Trivializable intersection)**.** Let Q be a connected component of the intersection of two modules M and N. For each point $x \in I_Q$, define

$$d_{triv}^{(M,N)}(x) = \max\{\mathrm{dl}(x, U(I_M))/2, \mathrm{dl}(x, L(I_N))/2)\}.$$

For $\delta \geq 0$, we say a point x is $\delta_{(M,N)}$-*trivializable* if $d_{triv}^{(M,N)}(x) < \delta$. We say an intersection component Q is $\delta_{(M,N)}$-*trivializable* if each point in I_Q is $\delta_{(M,N)}$-*trivializable* (Figure 1).

Following proposition discretizes the search for trivializability (see the full version [18] for a proof).

▶ **Proposition 21.** An intersection component Q is $\delta_{(M,N)}$-*trivializable* if and only if every vertex of Q is $\delta_{(M,N)}$-*trivializable*.

Recall that for two modules to be δ-interleaved, we need two families of linear maps satisfying both triangular commutativity and square commutativity. For a given δ, Theorem 23 below provides criteria which ensure that such linear maps exist. In our algorithm, we make sure that these criteria are verified.

Given an interval module M and the diagonal line Δ_x for any $x \in \bar{\mathbb{R}}^2$, there is a 1-dimensional persistence module $M|_{\Delta_x}$ which is the functor restricted on the poset $\mathbf{\Delta}_x$ as a subcategory of $\bar{\mathbb{R}}^2$. We call it a *1-dimensional slice* of M along Δ_x. Define

$$\delta^* = \inf_{\delta \in \bar{\mathbb{R}}} \{\delta : \forall x \in \bar{\mathbb{R}}^2, M|_{\Delta_x} \text{ and } N|_{\Delta_x} \text{ are } \delta\text{-interleaved}\}.$$

Proposition 22 follows from the observation that $\delta^* = \sup_{x \in \bar{\mathbb{R}}^2} \{d_I(M|_{\Delta_x}, N|_{\Delta_x})\}$.

▶ **Proposition 22.** For two interval modules M, N and $\delta \in \mathbb{R}^+$, we have $\delta > \delta^*$ if and only if there exist two families of linear maps $\phi = \{\phi_x : M_x \to N_{(x+\delta)}\}$ and $\psi = \{\psi_x : N_x \to M_{(x+\delta)}\}$ such that for each $x \in \bar{\mathbb{R}}^2$, the 1-dimensional slices $M|_{\Delta_x}$ and $N|_{\Delta_x}$ are δ-interleaved by the linear maps $\phi|_{\Delta_x}$ and $\psi|_{\Delta_x}$.

▶ **Theorem 23.** *Two interval modules M and N are δ-interleaved if and only if*
- $\delta > \delta^*$, *and*
- *each component of $I_M \cap I_{N_{\to\delta}}$ is either $(M, N_{\to\delta})$-valid or $\delta_{(M,N_{\to\delta})}$-trivializable, and each component of $I_{M_{\to\delta}} \cap N$ is either $(N, M_{\to\delta})$-valid or $\delta_{(N,M_{\to\delta})}$-trivializable.*

Proof. $\Longrightarrow$ direction: Suppose M and N are δ-interleaved. By definition, we have two families of linear maps $\{\phi_x\}$ and $\{\psi_x\}$ which satisfy both triangular and square commutativities. Let the morphisms between the two persistence modules constituted by these two families of linear maps be $\phi = \{\phi_x\}$ and $\psi = \{\psi_x\}$ respectively. By Proposition 22, we get the first part of the claim that $\delta > \delta^*$. For each intersection component Q of M and $N_{\to\delta}$ with interval $I := I_Q$, consider the restriction $\phi|_I$. By Proposition 12, $\phi|_I$ is constant, that is, $\phi|_I \equiv 0$ or $\nvDash$. If $\phi|_I \equiv \nvDash$, by Proposition 14, Q is $(M, N_{\to\delta})$-*valid*. If $\phi|_I \equiv 0$, by the triangular commutativity of ϕ, we have that $\rho^M_{x \to x+2\vec{\delta}} = \psi_{x+\vec{\delta}} \circ \phi_x = 0$ for

each point $x \in I$. That means $x + 2\vec{\delta} \notin I_M$. By Fact 18(i), $\mathrm{dl}(x, U(I_M))/2 < \delta$. Similarly, $\rho^N_{x-\vec{\delta}\to x+\vec{\delta}} = \phi_x \circ \psi_{x-\vec{\delta}} = 0 \implies x - \vec{\delta} \notin I_N$, which is the same as to say $x - 2\vec{\delta} \notin I_{N_{\to\delta}}$. By Fact 18(i), $\mathrm{dl}(x, L(I_{N_{\to\delta}}))/2 < \delta$. So $\forall x \in I$, we have $d_{triv}^{(M,N_{\to\delta})}(x) < \delta$. This means Q is $\delta_{(M,N_{\to\delta})}$-*trivializable*. Similar statement holds for intersection components of $M_{\to\delta}$ and N.

$\Longleftarrow$ direction: We construct two families of linear maps $\{\phi_x\}, \{\psi_x\}$ as follows: On the interval $I := I_{Q_i}$ of each intersection component Q_i of M and $N_{\to\delta}$, set $\phi|_I \equiv \nVdash$ if Q_i is $(M, N_{\to\delta})$-*valid* and $\phi|_I \equiv 0$ otherwise. Set $\phi_x \equiv 0$ for all x not in the interval of any intersection component. Similarly, construct $\{\psi_x\}$. Note that, by Proposition 14, $\phi := \{\phi_x\}$ is a morphism between M and $N_{\to\delta}$, and $\psi := \{\psi_x\}$ is a morphism between N and $M_{\to\delta}$. Hence, they satisfy the square commutativity. We show that they also satisfy the triangular commutativity. We claim that $\forall x \in I_M$, $\rho^M_{x\to x+2\vec{\delta}} = \nVdash \implies x + \vec{\delta} \in I_N$ and similar statement holds for I_N. From condition that $\delta > \delta^*$ and by Proposition 22, we know that there exist two families of linear maps satisfying triangular commutativity everywhere, especially on the pair of 1-dimensional persistence modules $M|_{\Delta_x}$ and $N|_{\Delta_x}$. From triangular commutativity we know that $x + \vec{\delta} \in I_N$ since otherwise one cannot construct a δ-interleaving between $M|_{\Delta_x}$ and $N|_{\Delta_x}$. Now for each $x \in I_M$ with $\rho^M_{x\to x+2\vec{\delta}} = \nVdash$, we have $\mathrm{dl}(x, U(I_M))/2 \geq \delta$ by Fact 18, and $x + \vec{\delta} \in I_N$ by our claim. This implies that $x \in I_M \cap I_{N_{\to\delta}}$ is a point in an interval of an intersection component Q_x of $M, N_{\to\delta}$ which is not $\delta_{(M,N_{\to\delta})}$-*trivializable*. Hence, it is $(M, N_{\to\delta})$-*valid* by the assumption. So, by our construction of ϕ on valid intersection components, $\phi_x = \nVdash$. Symmetrically, we have that $x + \vec{\delta} \in I_N \cap I_{M\to\delta}$ is a point in an interval of an intersection component of N and $M_{\to\delta}$ which is not $\delta_{(N,M_{\to\delta})}$-*trivializable* since $\mathrm{dl}(x + \vec{\delta}, L(I_M))/2 \geq \delta$. So by our construction of ψ on valid intersection components, $\psi_{x+\vec{\delta}} = \nVdash$. Then, we have $\rho^M_{x\to x+2\vec{\delta}} = \psi_{x+\vec{\delta}} \circ \phi_x$ for every nonzero linear map $\rho^M_{x\to x+2\vec{\delta}}$. The statement also holds for any nonzero linear map $\rho^N_{x\to x+2\vec{\delta}}$. Therefore, the triangular commutativity holds. ◀

Note that the above proof provides a construction of the interleaving maps for a specific δ if it exists. Furthermore, the interleaving distance $d_I(M, N)$ is the infimum of all δ satisfying the two conditions in the theorem, which means $d_I(M, N)$ is the infimum of all $\delta > \delta^*$ satisfying condition 2 in Theorem 23. Based on this observation, we propose a search algorithm for computing the interleaving distance $d_I(M, N)$ for interval modules M and N.

▶ **Definition 24** (Candidate set)**.** For two interval modules M and N, and for each point x in $I_M \cup I_N$, let

$$\begin{aligned} D(x) &= \{\mathrm{dl}(x, L(I_M)), \mathrm{dl}(x, L(I_N)), \mathrm{dl}(x, U(I_M)), \mathrm{dl}(x, U(I_N))\} \text{ and} \\ S &= \{d \mid d \in D(x) \text{ or } 2d \in D(x) \text{ for some vertex } x \in V(I_M) \cup V(I_N)\} \text{ and} \\ S_{\geq\delta} &:= \{d \mid d \geq \delta, d \in S\}. \end{aligned}$$

Algorithm Interleaving (output: $d_I(M, N)$, input: I_M and I_N with t vertices in total)

1. Compute the candidate set S and let ϵ be the smallest difference between any two numbers in S. /* $O(t)$ time */
2. Compute δ^*; Let $\delta = \delta^*$. /* $O(t)$ time */
3. Output δ after a binary search in $S_{\geq\delta^*}$ by following steps /* $O(\log t)$ probes */
   - let $\delta' = \delta + \epsilon$
   - Compute intersections $I_M \cap I_{N_{\to\delta'}}$ and $I_N \cap I_{M_{\to\delta'}}$. /* $O(t)$ time */
   - For each intersection component, check if it is valid or trivializable according to Theorem 23. /* $O(t)$ time */

In the above algorithm, the following generic task of computing *diagonal span* is performed for several steps. Let L and U be any two chains of vertical and horizontal edges that are both x- and y-monotone. Assume that L and U have at most t vertices. Then, for a set X of $O(t)$ points in L, one can compute the intersection of Δ_x with U for every $x \in X$ in $O(t)$ total time. The idea is to first compute by a binary search a point x in X so that Δ_x intersects U if at all. Then, for other points in X, traverse from x in both directions while searching for the intersections of the diagonal line with U in lock steps.

Now we analyze the complexity of the algorithm Interleaving. The candidate set, by definition, has only $2t$ values which can be computed in $O(t)$ time by the diagonal span procedure. Proposition 25 shows that δ^* is in S and can be determined by computing the one dimensional interleaving distances $d_I(M|_{\Delta_x}, N|_{\Delta_x})$ for diagonal lines passing through $O(t)$ vertices of I_M and I_N. This can be done in $O(t)$ time by diagonal span procedure. Once we determine δ^*, we search for $\delta = d_I(M, N)$ in the truncated set $S_{\delta \geq \delta^*}$ to satisfy the first condition of Theorem 23. Intersections between two polygons I_M and I_N bounded by x- and y-monotone chains can be computed in $O(t)$ time by a simple traversal of the boundaries. The validity and trivializability of each intersection component can be determined in time linear in the number of its vertices due to Proposition 19 and Proposition 21 respectively. Since the total number of intersection points is $O(t)$, validity check takes $O(t)$ time in total. The check for trivializability also takes $O(t)$ time if one uses the diagonal span procedure.

Proposition 25 below says that δ^* is determined by a vertex in I_M or I_N and $\delta^* \in S$. Its proof appears in the full version [18].

▶ **Proposition 25.** (i) $\delta^* = \max_{x \in V(I_M) \cup V(I_N)} \{d_I(M|_{\Delta_x}, N|_{\Delta_x})\}$, (ii) $\delta^* \in S$.

The correctness of the algorithm Interleaving already follows from Theorem 23 as long as the candidate set contains the distance $d_I(M, N)$. The following concept of stable intersections helps us to establish this result.

▶ **Definition 26** (Stable intersection)**.** Let Q be an intersection component of M and N. We say Q is stable if every intersection point $x \in I_Q \cap B(I_M) \cap B(I_N)$ is non-degenerate, that is, x is in the interior of two edges $e_1 \in E(I_M)$ and $e_2 \in E(I_N)$, and $e_1 \perp e_2$ at x.

From Proposition 42 and Corollary 43 in Appendix A of the full version [18], we have the following claim.

▶ **Proposition 27.** $d \notin S$ if and only if each intersection component of $M, N_{\to d}$, and $N_{\to d}, M$ is stable.

The main property of a stable intersection component Q of M and N is that if we shift one of the interval module, say N, to $N_{\to \epsilon}$ continuously for some small value $\epsilon \in \mathbb{R}^+$, the interval I_{Q^ϵ} of the intersection component Q^ϵ of M and $N_{\to \epsilon}$ changes continuously. Next proposition follows directly from the stability of intersection components.

▶ **Proposition 28.** For a stable intersection component Q of M and N, there exists a positive real $\delta \in \mathbb{R}^+$ so that the following holds:

For each $\epsilon \in (-\delta, +\delta)$, there exists a unique intersection component Q^ϵ of M and $N_{\to \epsilon}$ so that it is still stable and $I_{Q^\epsilon} \cap I_Q \neq \emptyset$. Furthermore, there is a bijection $\mu_\epsilon : V(I_Q) \to V(I_{Q^\epsilon})$ so that $\forall x \in V(I_Q)$, x and $\mu_\epsilon(x)$ are on the same horizontal, vertical, or diagonal line, and $d_\infty(\mu_\epsilon(x), x) = \epsilon$. We call the set $\{Q^\epsilon \mid \epsilon \in (-\delta, +\delta)\}$ a stable neighborhood of Q.

▶ **Corollary 29.** *For a stable intersection component Q, we have:*
(i) Q is (M, N)-valid iff each Q^ϵ in the stable neighborhood is $(M, N_{\to \epsilon})$-valid.
(ii) If Q is $d_{(M,N)}$-trivializable, then Q^ϵ is $(d + 2\epsilon)_{(M,N_{\to \epsilon})}$-trivializable.

Proof. (i): Let Q^ϵ be any intersection component in a stable neighborhood of Q. We know that if Q is (M,N)-valid, then $VL(I_Q) \subseteq L(I_M)$ and $VU(I_Q) \subseteq U(I_N)$. By Proposition 28, $\mu_\epsilon(VL(I_Q)) = VL(I_{Q^\epsilon}) \subseteq L(I_M)$ and $\mu_\epsilon(UL(I_Q)) = UL(I_{Q^\epsilon}) \subseteq L(I_{N\to\epsilon})$. So Q^ϵ is $(M, N_{\to\epsilon})$-valid. Other direction of the implication can be proved by switching the roles of Q and Q^ϵ in the above argument.

(ii): From Proposition 28, we have that $\forall x' \in V(I_{Q^\epsilon})$, there exists a point $x \in V(I_Q)$ so that x and x' are on some horizontal, vertical, or diagonal line (Δ_x), and $d_\infty(x,x') \le \epsilon$. Then, by Fact 18(ii), one observes

$$d_{triv}^{(M,N_{\to\epsilon})}(x) \le d_{triv}^{(M,N_{\to\epsilon})}(x') + \epsilon \le d_{triv}^{(M,N)}(x) + 2\epsilon < d + 2\epsilon.$$

Therefore, Q^ϵ is $(d+2\epsilon)_{(M,N_{\to\epsilon})}$-trivializable. ◀

▶ **Theorem 30.** $d_I(M,N) \in S$.

Proof. Suppose that $d = d_I(M,N) \notin S$. Let d^* be the largest value in S satisfying $d^* \le d$. Note that $d \in S$ if and only if $d = d^*$. Then, $d^* < d$ by our assumption that $d \notin S$.

By definition of interleaving distance, we have $\forall d' > d$, there is a d'-interleaving between M and N, and $\forall d'' < d$, there is no d''-interleaving between M and N. By Proposition 25(ii), one can see that $\delta^* \le d^* < d$. So, to get a contradiction, we just need to show that there exists d'', $d^* < d'' < d$, satisfying the condition 2 in Theorem 23.

Let Q be any intersection component of $M, N_{\to d}$ or $N, M_{\to d}$. Without loss of generality, assume Q is an intersection component of M and $N_{\to d}$. By Proposition 27, Q is stable. We claim that there exists some $\epsilon > 0$ such that $Q^{-\epsilon}$ is an intersection component of M and $N_{\to d-\epsilon}$ in a stable neighborhood of Q, and $Q^{-\epsilon}$ is either $(M, N_{\to d-\epsilon})$-valid or $(d-\epsilon)_{(M,N_{\to d-\epsilon})}$-trivializable.

Let $\epsilon > 0$ be small enough so that $Q^{+\epsilon}$ is a stable intersection component of M and $N_{\to d+\epsilon}$ in a stable neighborhood of Q. By Theorem 23, $Q^{+\epsilon}$ is either $(M, N_{\to(d+\epsilon)})$-valid or $(d+\epsilon)_{(M,N_{\to(d+\epsilon)})}$-trivializable. If $Q^{+\epsilon}$ is $(M, N_{\to(d+\epsilon)})$-valid, then by Corollary 29(i), any intersection component in a stable neighborhood of Q is valid, which means there exists $Q^{-\epsilon}$ that is $(M, N_{\to d-\epsilon})$-valid for some $\epsilon > 0$. Now assume $Q^{+\epsilon}$ is not $(M, N_{\to(d+\epsilon)})$-valid. Then, $\forall \epsilon > 0$, $Q^{+\epsilon}$ is $(M, N_{\to(d+\epsilon)})$-trivializable, By Proposition 21 and 29(ii), we have $\forall x \in V(I_Q)$, $d_{triv}^{(M,N_{\to d+\epsilon})}(x) < d + 3\epsilon$, $\forall \epsilon > 0$. Taking $\epsilon \to 0$, we get $\forall x \in V(I_Q)$, $d_{triv}^{(M,N_{\to d})}(x) \le d$. We claim that, actually, $\forall x \in V(I_Q)$, $d_{triv}^{(M,N_{\to d})}(x) < d$. If the claim were not true, some point $x \in V(I_Q)$ would exist so that $d_{triv}^{(M,N_{\to d})}(x) = d$. There are two cases. If $x \in V(I_M) \cup V(I_N)$, then obviously $d = d_{triv}^{(M,N_{\to d})}(x) \in S$ contradicting $d \neq d^*$. The other case is that x is the intersection point of two perpendicular edges $e_1 \in E(I_M)$ and $e_2 \in E(I_N)$ since Q is a stable intersection component. But, then x and $\pi_L(x)$ are always on two parallel edges where L is either $U(I_M)$ or $L(I_N)$. By Proposition 41(ii) in [18], we have $d = d^*$, reaching a contradiction. Now by our claim and Proposition 21, Q is $d_{(M,N_{\to d})}$-trivializable where $d > d^* \ge \max_{x\in V(I_Q)}\{d_{triv}^{(M,N_{\to d})}(x)\}$. Let $\delta = d - d^*$ and $\epsilon = \delta/4$. Since $d - \epsilon = d - \delta/4 > d - \delta/2 = d - \delta + 2\cdot\delta/4 = d^* + 2\epsilon$ and $d^* \ge \max_{x\in V(I_Q)}\{d_{triv}^{(M,N_{\to d})}(x)\}$, we have $d > d^*$ and $d - \epsilon > \max_{x\in V(I_Q)}\{d_{triv}^{(M,N_{\to d})}(x)\} + 2\epsilon$. Therefore, by Corollary 21, $Q^{-\epsilon}$ is $(d-\epsilon)_{(M,N_{\to d-\epsilon})}$-trivializable.

The above argument shows that there exists a d''-interleaving where $d'' = d - \epsilon < d$, reaching a contradiction. ◀

4 A lower bound on d_I

In this section we propose a distance between two persistence modules that bounds the interleaving distance from below. This distance is defined for n-D modules and not necessarily only for 2-D modules. It is based on dimensions of the vectors involved with the two modules and is efficiently computable.

Let $[n] = \{1, 2, \ldots, n\}$ be the set of all the integers from 1 to n. Let $\binom{[n]}{k} = \{s \subseteq [n] : |s| = k\}$ be the set of all subset in $[n]$ with cardinality k.

▶ **Definition 31.** For a right continuous function $f : \mathbb{R}^n \to \mathbb{Z}$, define the *differential* of f to be $\Delta f : \mathbb{R}^n \to \mathbb{Z}$ where

$$\Delta f(x) = \sum_{k=0}^{n} (-1)^k \cdot \sum_{s \in \binom{[n]}{k}} \lim_{\epsilon \to 0_+} f(x - \epsilon \cdot \sum_{i \in s} e_i)$$

Note that for $k = 0$, $\sum_{s \in \binom{[n]}{k}} \lim_{\epsilon \to 0_+} f(x - \epsilon \cdot \sum_{i \in s} e_i) = f(x)$. We say f is *nice* if the support $supp(\Delta f)$ is finite and $supp(f) \subseteq \{x \,|\, x \geq \vec{a}\}$ for some $a \in \mathbb{R}$.

The differential Δf is a function recording the change of function values of f at each point, especially at 'jump points'. For $n = 1$, $\Delta f(x) = f(x) - \lim_{\epsilon \to 0_+} f(x - \epsilon)$. For $n = 2$, which is the case we deal with, we have

$$\Delta f(x) = f(x) - \lim_{\epsilon \to 0_+} f(x - (\epsilon, 0)) - \lim_{\epsilon \to 0_+} f(x - (0, \epsilon)) + \lim_{\epsilon \to 0_+} f(x - (\epsilon, \epsilon)).$$

See Figure 2 and 3 for illustrations in 1- and 2-D cases respectively.

▶ **Proposition 32.** For a nice function f, $f(x) = \sum_{y \leq x} \Delta f(y)$.

For a proof see the full version [18].

We also define $\Delta f_+ = \max\{\Delta f, 0\}$, $\Delta f_- = \min\{\Delta f, 0\}$ and $f_{\Sigma+}(x) = \sum_{y \leq x} \Delta f_+(y)$, $f_{\Sigma-}(x) = \sum_{y \leq x} \Delta f_-(y)$. Note that $f_{\Sigma+} \geq 0$, $f_{\Sigma-} \leq 0$, and are both monotonic functions. By definition and property of Δf, we have $f = f_{\Sigma+} + f_{\Sigma-}$.

▶ **Definition 33.** For any $\delta > 0$, we define the δ-extension of f as $f^{+\delta} = f_{\Sigma+}(x+\delta) + f_{\Sigma-}(x-\delta)$. Similarly we define the δ-shrinking of f as $f^{-\delta} = f_{\Sigma-}(x + \delta) + f_{\Sigma+}(x - \delta)$ (see Figure 2).

Proposition 34 below follows from the definition.

▶ **Proposition 34.** For any $\delta > 0 \in \mathbb{R}$, we have $f^{\pm\delta}(x) = f(x \mp \delta) + \sum_{y \leq x \pm \delta, y \not\leq x \mp \delta} \Delta f_\pm(y)$.

That is to say, for any $\delta \in \mathbb{R}$, the extended (shrunk) function f^δ, can be computed by adding to $f(x - |\delta|)$ the positive (negative) difference values of Δf in $(x - |\delta|, x + |\delta|]$. From this, it follows:

▶ **Corollary 35.** *Given $0 \leq \delta \leq \delta' \in \mathbb{R}$, we have $f^{+\delta} \leq f^{+\delta'}$ and $f^{-\delta} \geq f^{-\delta'}$.*

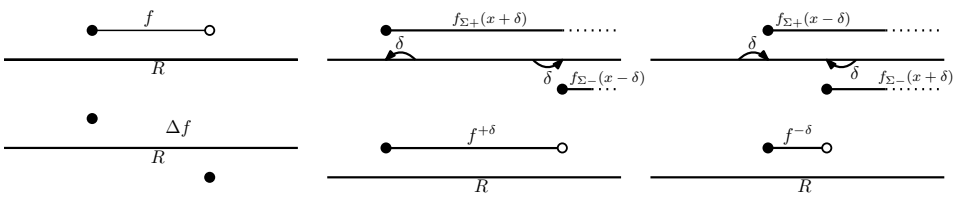


Figure 2 A nice function and its differential (left), its δ-extension (middle), δ-shrinking (right).

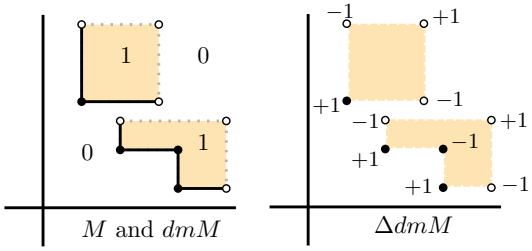


Figure 3 Dimension function (left), its differential is non-zero only at vertices (right).

▶ **Definition 36.** For any two nice functions $f, g : \mathbb{R}^n \to \mathbb{Z}$ and $\delta \geq 0$, we say f, g are within δ-extension, denoted as $f_{\leftarrow\delta\rightarrow}g$, if $f \leq g^{+\delta}$ and $g \leq f^{+\delta}$. Similarly, we say f, g are within δ-shrinking, denoted as $f_{\rightarrow\delta\leftarrow}g$, if $f \geq g^{-\delta}$ and $g \geq f^{-\delta}$.

Let d_+, d_-, d_0 be defined as follows on the space of all nice real-valued functions on $\mathbb{R}^n$:

$$d_-(f,g) = \inf_\delta \{\delta \mid f_{\rightarrow\delta\leftarrow}g\}, \quad d_+(f,g) = \inf_\delta \{\delta \mid f_{\leftarrow\delta\rightarrow}g\}, \quad d_0(f,g) = \min(d_-, d_+)$$

One can verify that d_0 is indeed a distance function. Also, note that when $f, g \geq 0$ (for example, f, g are dimension functions as defined below), we have $d_- \leq d_+$, hence $d_0 = d_-$. It seems that the definition of d_- has a similar connotation as the erosion distance defined by Patel [25] in 1-D case.

4.1 Dimension distance

Given a persistence module M, let the *dimension function* $\mathrm{dm}M : \mathbb{R}^n \to Z$ be defined as $\mathrm{dm}M(x) = dim(M_x)$. The distance $d_0(dmM, dmN)$ for two modules M and N is called the *dimension distance*. Our main result in theorem 38 is that this distance is stable with respect to the interleaving distance and thus provides a lower bound for it.

▶ **Definition 37.** A persistence module M is nice if there exists a value $\epsilon_0 \in \mathbb{R}^+$ so that for every $\epsilon < \epsilon_0$, each linear map $\rho^M_{x\to x+\vec{\epsilon}} : M_x \to M_{x+\vec{\epsilon}}$ is either injective or surjective (or both).

For example, a finitely presented persistence module generated by a simplicial filtration defined on a grid with at most one additional simplex being introduced between two adjacent grid points satisfies this nice condition above.

▶ **Theorem 38.** *For nice persistence modules M and N, $d_0(\mathrm{dm}M, \mathrm{dm}N) \leq d_I(M, N)$.*

Proof. Let $d_I(M, N) = \delta$. There exists δ-interleaving, $\phi = \{\phi_x\}, \psi = \{\psi_x\}$ which satisfy both triangular and square commutativity. We claim $(\mathrm{dm}M)^{-\delta} \leq \mathrm{dm}N$ and $(\mathrm{dm}N)^{-\delta} \leq \mathrm{dm}M$.

Let $x \in \mathbb{R}^n$ be any point. By Proposition 34, we know that $(\mathrm{dm}M)^{-\delta}(x) = \mathrm{dm}M(x - \delta) + \sum_{y\leq x+\delta, y\not\leq x-\delta}(\Delta\mathrm{dm}N_-)(y)$. If $\mathrm{dm}M(x-\delta) \leq \mathrm{dm}N(x)$, then we get $(\mathrm{dm}M)^{-\delta}(x) \leq \mathrm{dm}M(x-\delta) \leq \mathrm{dm}N(x)$, because $\sum_{y\leq x+\delta, y\not\leq x-\delta}(\Delta\mathrm{dm}N_-)(y) \leq 0$.

Now assume $\mathrm{dm}M(x-\delta) > \mathrm{dm}N(x)$. From triangular commutativity, we have $rank(\psi_x \circ \phi_{x-\vec{\delta}}) = rank(\rho^M_{x-\vec{\delta}\to x+\vec{\delta}})$, which gives $\dim(im(\rho^M_{x-\vec{\delta}\to x+\vec{\delta}})) \leq \dim(im(\phi_{x-\vec{\delta}})) \leq \mathrm{dm}N(x)$.

There exists a collection of linear maps $\{\rho_i : M_{x_i} \to M_{x_{i+1}}\}_{i=0}^k$ such that $\rho^M_{x-\vec{\delta}\to x+\vec{\delta}} = \rho_k \circ \rho_{k-1} \circ \ldots \circ \rho_1 \circ \rho_0$ and each ρ_i is either injective or surjective. Let $\mathrm{im}_i = im(\rho_i \circ \ldots \circ \rho_0)$. Note that $\mathrm{im}_k = im(\rho^M_{x-\vec{\delta}\to x+\vec{\delta}})$. Let $\epsilon_i = \dim(\mathrm{im}_i) - \dim(\mathrm{im}_{i-1})$. Then note that $\epsilon_i = 0$ if ρ_i is injective and $\dim(\mathrm{im}_k) - \dim(M_{x_0}) = \sum_{i=1}^k \epsilon_i$. Since $\dim(\mathrm{im}_k) - \dim(M_{x_0}) < 0$, there exists a collection of ρ_{i_j}'s such that $\epsilon_{i_j} < 0$. This means these ρ_{i_j}'s are non-isomorphic

surjective linear maps with $\dim(M_{x_{i_j}}) - \dim(M_{x_{i_j-1}}) < 0$. By definition of $\Delta\mathrm{dm}$, this means that, for each pair (x_{i_j-1}, x_{i_j}), there exists a collection $y_1, y_2, \ldots$ such that $y_l \leq x_{i_j}, y_l \not\leq x_{i_j-1}$ and $\sum_l (\Delta\mathrm{dm}M)_-(y_l) \leq \epsilon_{i_j}$. All these y's also satisfy that $y \leq x + \vec{\delta}, y \not\leq x - \vec{\delta}$. So,

$$\sum_{y \leq x+\delta, y \not\leq x-\delta} (\Delta\mathrm{dm}M_-)(y) \leq \sum_j \epsilon_j = \dim(\mathrm{im}_k) - \dim(M_{x_0}) \leq \dim(N_x) - \dim(M_{x-\vec{\delta}}),$$

which gives $(\mathrm{dm}M)^{-\delta}(x) \leq \mathrm{dm}N(x)$. Similarly, we can show $(\mathrm{dm}N)^{-\delta}(x) \leq \mathrm{dm}M(x)$. ◀

Notice that for dimension functions which are always non-negative, we have $d_0 = d_-$. It may seem that we could have avoided introducing d_+ altogether. But, since nice functions also include negative valued functions, one can verify that d_+ plays the same role for such functions as d_- does for non-negative ones. Then to make d_0 a distance on the space of all nice functions, one needs to define it as the minimum of both d_+ and d_-. For dimension functions, d_+ is not necessarily bounded above by d_I.

4.2 Computation of d_0

For computational purpose, assume that two input persistence modules M and N are finite in that they are functors on the subcategory $\{1, \ldots, k\}^n \subset \mathbb{R}^n$ and the dimension functions $f := \mathrm{dm}M$, $g := \mathrm{dm}N$ have been given as input on an n-dimensional k-ary grid.

First, for the dimension functions f, g, we compute $\Delta f, \Delta g, \Delta f_\pm, \Delta g_\pm, f_\pm, g_\pm$ in $O(k^2)$ time. By Proposition 34, for any $\delta \in \mathbb{Z}^+$, we can also compute $f^{\pm\delta}$, $g^{\pm\delta}$ in $O(k^2)$ time. Then we can apply the binary search to find the minimal value δ within a bounded region such that f, g are within δ-extension or δ-shrinking. This takes $O(\log k)$ time. So the entire computation takes $O(k^2 \log k)$ time.

5 Conclusions

In this paper, we presented an efficient algorithm to compute the bottleneck distance of two 2-D persistence modules given by indecomposables that may have non-constant complexity. No such algorithm for such case is known. Making the algorithm more efficient will be one of our future goals. Extending the algorithm or its modification to larger classes of modules such as the n-D modules or exact pfd bi-modules considered in [14] will be interesting. The definition of valid and trivializable intersection component and Theorem 21 can be extended easily to n-D modules. So is the algorithm– possibly with sacrificing some of the efficiencies. But, further work is necessary to establish the correctness of the algorithm for this general case.

The assumption of nice modules for dimension distance d_0 is needed so that the dimension function, which is a weaker invariant compared to the rank invariants or barcodes in one dimensional case, provides meaningful information without ambiguity. There are cases where the dimension distance can be larger than interleaving distance if the assumption of nice modules is dropped. Of course, one can adjust the definition of dimension distance to incorporate more information so that it remains bounded from above by the interleaving distance.

References

1 Michael Atiyah. On the krull-schmidt theorem with application to sheaves. *Bulletin de la Société Mathématique de France*, 84:307–317, 1956. URL: `http://eudml.org/doc/86907`.

2 Ulrich Bauer and Michael Lesnick. Induced matchings of barcodes and the algebraic stability of persistence. In *Proceedings of the Thirtieth Annual Symposium on Computational Geometry*, SOCG'14, pages 355:355–355:364, 2014.

3 Silvia Biasotti, Andrea Cerri, Patrizio Frosini, and Daniela Giorgi. A new algorithm for computing the 2-dimensional matching distance between size functions. *Pattern Recognition Letters*, 32(14):1735–1746, 2011.

4 Håvard Bjerkevik. Stability of higher-dimensional interval decomposable persistence modules. *arXiv preprint arXiv:1609.02086*, 2016.

5 Håvard Bjerkevik and Magnus Botnan. Computational complexity of the interleaving distance. *arXiv preprint arXiv:1712.04281*, 2017.

6 Magnus Botnan, Justin Curry, and Elizabeth Munch. The poset interleaving distance, 2016. URL: `https://jointmathematicsmeetings.org/amsmtgs/2180_abstracts/1125-55-1151.pdf`.

7 Magnus Botnan and Michael Lesnick. Algebraic stability of zigzag persistence modules. *arXiv preprint arXiv:1604.00655*, 2016.

8 Peter Bubenik and Jonathan Scott. Categorification of persistent homology. *Discrete & Computational Geometry*, 51(3):600–627, 2014.

9 Gunnar Carlsson and Afra Zomorodian. The theory of multidimensional persistence. *Discrete & Computational Geometry*, 42(1):71–93, Jul 2009.

10 Andrea Cerri, Barbara Di Fabio, Massimo Ferri, Patrizio Frosini, and Claudia Landi. Betti numbers in multidimensional persistent homology are stable functions. *Mathematical Methods in the Applied Sciences*, 36(12):1543–1557, 2013.

11 Andrea Cerri and Patrizio Frosini. A new approximation algorithm for the matching distance in multidimensional persistence. Technical report, February 2011. URL: `http://amsacta.unibo.it/2971`.

12 Frédéric Chazal, David Cohen-Steiner, Marc Glisse, Leonidas Guibas, and Steve Oudot. Proximity of persistence modules and their diagrams. In *Proceedings of the Twenty-fifth Annual Symposium on Computational Geometry*, SCG '09, pages 237–246, 2009.

13 Frédéric Chazal, Vin de Silva, Marc Glisse, and Steve Oudot. The structure and stability of persistence modules. *arXiv preprint arXiv:1207.3674*, 2012.

14 Jérémy Cochoy and Steve Oudot. Decomposition of exact pfd persistence bimodules. *arXiv preprint arXiv:1605.09726*, 2016.

15 David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. *Discrete & Computational Geometry*, 37(1):103–120, 2007.

16 William Crawley-Boevey. Decomposition of pointwise finite-dimensional persistence modules. *Journal of Algebra and Its Applications*, 14(05):1550066, 2015. `doi:10.1142/S0219498815500668`.

17 Vin de Silva, Elizabeth Munch, and Amit Patel. Categorified reeb graphs. *Discrete & Computational Geometry*, 55(4):854–906, Jun 2016.

18 Tamal Dey and Cheng Xin. Computing bottleneck distance for 2-d interval decomposable modules, 2018. URL: `http://arxiv.org/abs/1803.02869`.

19 Herbert Edelsbrunner and John Harer. *Computational Topology: An Introduction*. Applied Mathematics. American Mathematical Society, 2010.

20 Michael Kerber, Dmitriy Morozov, and Arnur Nigmetov. Geometry helps to compare persistence diagrams. *Journal of Experimental Algorithmics (JEA)*, 22(1):1–4, 2017.

21 Claudia Landi. The rank invariant stability via interleavings. *arXiv preprint arXiv:1412.3374*, 2014.

22 Michael Lesnick. The theory of the interleaving distance on multidimensional persistence modules. *Foundations of Computational Mathematics*, 15(3):613–650, 2015.

23 Klaus Lux and Magdolna Szőke. Computing decompositions of modules over finite-dimensional algebras. *Experimental Mathematics*, 16(1):1–6, 2007.

24 Steve Oudot. *Persistence theory: from quiver representations to data analysis*, volume 209. American Mathematical Society, 2015.

25 Amit Patel. Generalized persistence diagrams. *arXiv preprint arXiv:1601.03107*, 2016.

26 Carry Webb. Decomposition of graded modules. *Proc. American Math. Soc.*, 94(4):565–571, 1985.

Structure and Generation of Crossing-Critical Graphs

Zdeněk Dvořák[1]
Charles University, Prague, Czech Republic
rakdver@iuuk.mff.cuni.cz
https://orcid.org/0000-0002-8308-9746

Petr Hliněný[2]
Faculty of Informatics, Masaryk University, Brno, Czech Republic
hlineny@fi.muni.cz
https://orcid.org/0000-0003-2125-1514

Bojan Mohar[3]
Department of Mathematics, Simon Fraser University, Burnaby, BC V5A 1S6, Canada
mohar@sfu.ca
https://orcid.org/0000-0002-7408-6148

Abstract

We study c-crossing-critical graphs, which are the minimal graphs that require at least c edge-crossings when drawn in the plane. For $c = 1$ there are only two such graphs without degree-2 vertices, K_5 and $K_{3,3}$, but for any fixed $c > 1$ there exist infinitely many c-crossing-critical graphs. It has been previously shown that c-crossing-critical graphs have bounded path-width and contain only a bounded number of internally disjoint paths between any two vertices. We expand on these results, providing a more detailed description of the structure of crossing-critical graphs. On the way towards this description, we prove a new structural characterisation of plane graphs of bounded path-width. Then we show that every c-crossing-critical graph can be obtained from a c-crossing-critical graph of bounded size by replicating bounded-size parts that already appear in narrow "bands" or "fans" in the graph. This also gives an algorithm to generate all the c-crossing-critical graphs of at most given order n in polynomial time per each generated graph.

2012 ACM Subject Classification Theory of computation → Computational geometry, Mathematics of computing → Graphs and surfaces

Keywords and phrases crossing number, crossing-critical, path-width, exhaustive generation

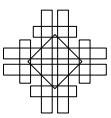
Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.33

Related Version A full version of this paper is available at arXiv:1803.01931.

Acknowledgements We would like to thank the anonymous referees for helpful comments on this submission.

[1] Z.D. was supported by the Center of Excellence – Institute for Theoretical Computer Science, Prague, project P202/12/G061 of the Czech Science Foundation.
[2] P.H. was supported by the Center of Excellence – Institute for Theoretical Computer Science, Brno, project P202/12/G061 of the Czech Science Foundation.
[3] B.M. was supported in part by the NSERC Discovery Grant R611450 (Canada), by the Canada Research Chairs program, and by the Research Project J1-8130 of ARRS (Slovenia).

34th International Symposium on Computational Geometry (SoCG 2018).
Editors: Bettina Speckmann and Csaba D. Tóth; Article No. 33; pp. 33:1–33:14



Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

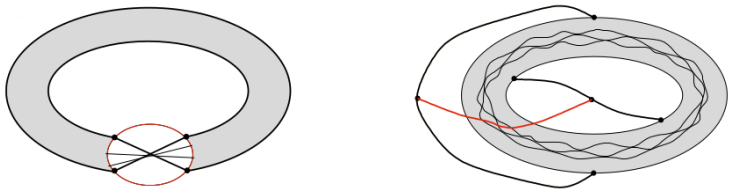


Figure 1 A schematic illustration of two basic methods of constructing crossing-critical graphs.

1 Introduction

Minimizing the number of edge-crossings in a graph drawing in the plane (the *crossing number* of the graph, cf. Definition 2.1) is considered one of the most important attributes of a "nice drawing" of a graph, and this question has found numerous other applications (for example, in VLSI design [12] and in discrete geometry [18]). Consequently, a great deal of research work has been invested into understanding what forces the graph crossing number to be high. There exist strong quantitative lower bounds, such as the famous Crossing Lemma [1, 12]. However, the quantitative bounds show their strength typically in dense graphs, and hence they do not shed much light on the structural properties of graphs of high crossing number.

The reasons for sparse graphs to have many crossings in any drawing are structural – there is a lot of "nonplanarity" in them. These reasons can be understood via corresponding minimal obstructions, the so called *c-crossing-critical* graphs (cf. Section 2 and Definition 2.2), which are the subgraph-minimal graphs that require at least c crossings. There are only two 1-crossing-critical graphs without degree-2 vertices, the Kuratowski graphs K_5 and $K_{3,3}$, but it has been known already since Širáň's [19] and Kochol's [11] constructions that the structure of c-crossing-critical graphs is quite rich and non-trivial for any $c \geq 2$. Already the first nontrivial case of $c = 2$ shows a dramatic increase in complexity of the problem. Yet, Bokal, Oporowski, Richter, and Salazar recently succeeded in obtaining a full description [3] of all the 2-crossing-critical graphs up to finitely many "small" exceptions.

To our current knowledge, there is no hope of extending the explicit description from [3] to any value $c > 2$. We, instead, give for any fixed positive integer c an asymptotic structural description of all sufficiently large c-crossing-critical graphs.

Contribution outline. We refer to subsequent sections for the necessary formal concepts. On a high level of abstraction, our contribution can be summarized as follows:

1. There exist three kinds of local arrangements – a crossed band of uniform width, a twisted band, or a twisted fan – such that any optimal drawing of a sufficiently large c-crossing-critical graph contains at least one of them.
2. There are well-defined local operations (replacements) performed on such bands or fans that can reduce any sufficiently large c-crossing-critical graph to one of finitely many base c-crossing-critical graphs.
3. A converse – a well-defined bounded-size expansion operation – can be used to iteratively construct each c-crossing-critical graph from a c-crossing-critical graph of bounded size. This yields a way to enumerate all the c-crossing-critical graphs of at most given order n in polynomial time per each generated graph. More precisely, the total runtime is $O(n)$ times the output size.

To give a closer (but still informal) explanation of these points, we should review some of the key prior results. First, the infinite 2-crossing-critical family of Kochol [11] explicitly showed one basic method of constructing crossing-critical graphs – take a sequence of suitable small planar graphs (called *tiles*, cf. Section 3), concatenate them naturally into a plane strip and join the ends of this strip with the *Möbius twist*. See Figure 1. Further constructions of this kind can be found, e.g., in [2, 14, 16]. In fact, [3] essentially claims that such a Möbius twist construction is the only possibility for $c = 2$; there, the authors give an explicit list of 42 tiles which build in this way all the 2-crossing-critical graphs up to finitely many exceptions.

The second basic method of building crossing-critical graphs was invented later by Hliněný [9]; it can be roughly described as constructing a suitable planar strip whose ends are now joined without a twist (i.e., making a cylinder), and adding to it a few edges which then have to cross the strip. See again Figure 1 for an illustration. Furthermore, diverse crossing-critical constructions can easily be combined together using so called *zip product* operation of Bokal [2] which preserves criticality. To complete the whole picture, there exists a third, somehow mysterious method of building c-crossing-critical graphs (for sufficiently high values of c), discovered by Dvořák and Mohar in [5]. The latter can be seen as a degenerate case of the Möbius twist construction, such that the whole strip shares a central high-degree vertex, and we skip more details till the technical parts of this paper.

As we will see, the three above sketched construction methods roughly represent the three kinds of local arrangements mentioned in point (1). In a sense, we can thus claim that no other method (than the previous three) of constructing infinite families of c-crossing-critical graphs is possible, for any fixed c. Moving on to point (2), we note that all three mentioned construction methods involve long (and also "thin") planar strips, or *bands* as subgraphs (which degenerate into *fans* in the third kind of local arrangements; cf. Definition 3.1). We will prove, see Corollary 3.6, that such a long and "thin" planar band or fan must exist in any sufficiently large c-crossing-critical graph, and we analyse its structure to identify elementary connected tiles of bounded size forming the band. We then argue that we can reduce repeated sections of the band while preserving c-crossing-criticality. Regarding point (3), the converse procedure giving a generic bounded-size expansion operation on c-crossing-critical graphs is described in Theorem 4.9 (for a quick illustration, the easiest case of such an expansion operation is edge subdivision, that is replacing an edge with a path, which clearly preserves c-crossing-criticality).

Paper organization. After giving the definitions and preliminary results about crossing-critical graphs in Section 2, we show a new structural characterisation of plane graphs of bounded path-width which forms the cornerstone of our paper in Section 3. Then, in Section 4, we deal with the structure and reductions / expansions of crossing-critical graphs, presenting our main results. In Section 5 we outline the technical steps leading to our cornerstone characterisation from Section 3. Some final remarks are presented in Section 6.

Due to restrictions on the length of the paper, some technical details and proofs of our statements are left for the full paper. Statements whose proofs are in the full paper are marked with **(*)**.

2 Graph drawing and the crossing number

In this paper, we consider multigraphs by default, even though we could always subdivide parallel edges (with a slight adjustment of definitions) in order to make our graphs simple. We follow basic terminology of topological graph theory, see e.g. [13].

A *drawing* of a graph G in the plane is such that the vertices of G are distinct points and the edges are simple curves joining their end vertices. It is required that no edge passes through a vertex, and no three edges cross in a common point. A *crossing* is then an intersection point of two edges other than their common end. A drawing without crossings in the plane is called a *plane drawing* of a graph, or shortly a *plane graph*. A graph having a plane drawing is *planar*.

The following are the core definitions of our research.

▶ **Definition 2.1** (crossing number)**.** The *crossing number* $\mathrm{cr}(G)$ of a graph G is the minimum number of crossings of edges in a drawing of G in the plane.

▶ **Definition 2.2** (crossing-critical)**.** Let c be a positive integer. A graph G is *c-crossing-critical* if $cr(G) \geq c$, but every proper subgraph G' of G has $cr(G') < c$.

Furthermore, suppose G is a graph drawn in the plane with crossings. Let G' be the plane graph obtained from this drawing by replacing the crossings with new vertices of degree 4. We say that G' is the plane graph associated with the drawing, shortly the *planarization of* G, and the new vertices are the *crossing vertices* of G'.

Preliminaries. Structural properties of crossing-critical graphs have been studied for more than two decades, and we now briefly review some of the previous important results which we shall use. First, we remark that a c-crossing-critical graph may have no drawing with only c crossings (examples exist already for $c = 2$). Richter and Thomassen [15] proved the following upper bound:

▶ **Theorem 2.3** ([15])**.** *Every c-crossing-critical graph has a drawing with at most $\lceil 5c/2+16 \rceil$ crossings.*

Interestingly, although the bound of Theorem 2.3 sounds rather weak and we do not know any concrete examples requiring more than $c + \mathcal{O}(\sqrt{c})$ crossings, the upper bound has not been improved for more than two decades. We not only use this important upper bound, but also hope to be able to improve it in the future using our results.

Our approach to dealing with "long and thin" subgraphs in crossing-critical graphs relies on the folklore structural notion of *path-width* of a graph, which we recall in Definition 3.4. Hliněný [7] proved that c-crossing-critical graphs have path-width bounded in terms of c, and he and Salazar [8] showed that c-crossing-critical graphs can contain only a bounded number of internally disjoint paths between any two vertices.

▶ **Theorem 2.4** ([7])**.** *Every c-crossing-critical graph has path-width (cf. Definition 3.4) at most $\lceil 2^{6(72 \log_2 c+248)c^3+1} \rceil$.*

Another useful concept for this work is that of *nests* in a drawing of a graph (cf. Definition 3.3), implicitly considered already in previous works [7, 8], and explicitly defined by Hernandez-Velez et al. [6] who concluded that no optimal drawing of a c-crossing-critical graph can contain a 0-, 1-, or 2-nest of large depth compared to c.

Lastly, we remark that by trivial additivity of the crossing number over blocks, we may (and will) restrict our attention only to 2-*connected crossing-critical graphs*. We formally argue as follows. For $c, \delta > 0$, let us say a graph is (c, δ)-*crossing-critical* if it has crossing number *exactly* c and all proper subgraphs have crossing number at most $c - \delta$.

▶ **Proposition 2.5** (folklore)**.** *A graph H is c-crossing-critical if and only if there exist positive integers $c_1, \ldots, c_b$ and δ such that $c \leq c_1 + \cdots + c_b \leq c + \delta - 1$, H has exactly b 2-connected blocks $H_1, \ldots, H_b$, and the block H_i is (c_i, δ)-crossing-critical for $i = 1, \ldots, b$.*

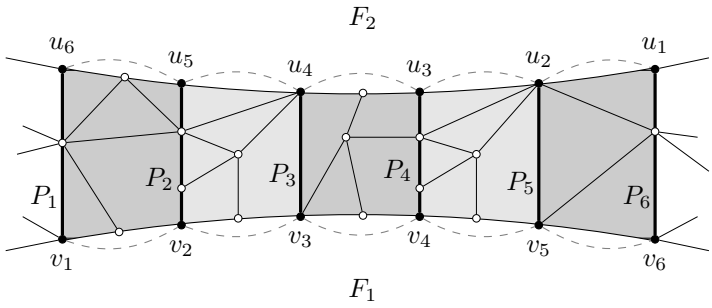


Figure 2 An example of paths $P_1, \ldots, P_6$ (bold lines) forming an (F_1, F_2)-band of length 6, cf. Definition 3.1. The five tiles of this band, as in Definition 3.2, are shaded in grey and the dashed arcs represent α_i and α'_i from that definition.

Hence, strictly respecting Proposition 2.5, we should actually study 2-connected (c, δ)-crossing-critical graphs. To keep the presentation simpler, we stick with c-crossing-critical graphs, but we remark that our results also hold in the more refined setting.

3 Structure of plane tiles

The proof of our structural characterisation of crossing-critical graphs can be roughly divided into two main parts. The first one, presented in this section (leaving technical prerequisites for later Section 5), establishes the existence of specific plane bands (resp. fans) and their tiles in crossing-critical graphs. The second part will then, in Section 4, closely analyse these bands and tiles. Unlike a more traditional "bottom-up" approach to tiles in crossing number research (e.g., [3]), we define tiles and deal with them "top-down", i.e., describing first plane bands or fans and then identifying tiles as their small elementary parts. Our key results are summarized below in Theorem 3.5 and Corollary 3.6.

▶ **Definition 3.1** (band and fan)**.** Let G be a 2-connected plane graph. Let F_1 and F_2 be distinct faces of G and let $v_1, v_2, \ldots, v_m$, and $u_1, u_2, \ldots, u_m$ be some of the vertices incident with F_1 and F_2, respectively, listed in the cyclic order along the faces. If $P_1, \ldots, P_m$ are pairwise vertex-disjoint paths in G such that P_i joins v_i with u_{m+1-i}, for $1 \leq i \leq m$, then we say that $(P_1, \ldots, P_m)$ forms an (F_1, F_2)-*band of length* m. Note that P_i may consist of only one vertex $v_i = u_{m+1-i}$.

Let F_1 and $v_1, v_2, \ldots, v_m$ be as above. If u is a vertex of G and $P_1, \ldots, P_m$ are paths in G such that P_i joins v_i with u, for $1 \leq i \leq m$, and the paths are pairwise vertex-disjoint except for their common end u, then we say that $(P_1, \ldots, P_m)$ forms an (F_1, u)-*fan of length* m. The (F_1, u)-fan is *proper* if u is not incident with F_1.

▶ **Definition 3.2** (tiles and support)**.** Let $(P_1, \ldots, P_m)$ be either an (F_1, F_2)-band or an (F_1, u)-fan of length $m \geq 3$. For $1 \leq i \leq m-1$, let α_i be an arc between v_i and v_{i+1} drawn inside F_1, and let α'_i be an arc drawn between u_i and u_{i+1} in F_2 in the case of the band; α'_i are null when we are considering a fan. Furthermore, choose the arcs to be internally disjoint. Let θ_i be the closed curve consisting of P_i, α_i, P_{i+1}, and α'_{m-i}. Let λ_i be the connected part of the plane minus θ_i that contains none of the paths P_j $(1 \leq j \leq m)$ in its interior. The subgraphs of G drawn in the closures of $\lambda_1, \ldots, \lambda_{m-1}$ are called *tiles* of the band or fan (and the tile of λ_i includes $P_i \cup P_{i+1}$ by this definition). The union of these tiles is the *support* of the band or fan.

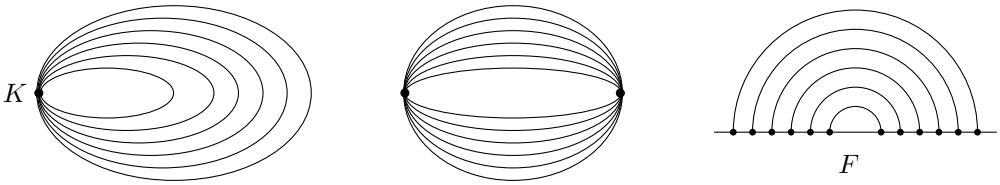


Figure 3 An illustration of Definition 3.3: a 1-nest, a 2-nest, and an F-nest, each of depth 6.

▶ **Definition 3.3** (nests). Let G be a 2-connected plane graph. For an integer $k \geq 0$, a *k-nest* in G of *depth* m is a sequence $(C_1, C_2, \ldots, C_m)$ of pairwise edge-disjoint cycles such that for some set K of k vertices and for every $i < j$, the cycle C_i is drawn in the closed disk bounded by C_j and $V(C_i) \cap V(C_j) = K$.

Let F be a face of G and let v_1, v_2, …, v_{2m} be some of the vertices incident with F listed in the cyclic order along the face. Let P_1, …, P_m be pairwise vertex-disjoint paths in G such that P_i joins v_i with v_{2m+1-i}, for $1 \leq i \leq m$. Then, we say that $(P_1, \ldots, P_m)$ forms an *F-nest of depth m*. Similarly, let v_1, v_2, …, v_m, u be some of the vertices incident with F, let P_1, …, P_m be paths in G such that P_i joins v_i with u, for $1 \leq i \leq m$, and the paths intersect only in u. Then, we say that $(P_1, \ldots, P_m)$ form a *degenerate F-nest of depth m*.

See Figure 3. Note that degenerate F-nests are the same as non-proper (F, u)-fans.

Our cornerstone claim, interesting on its own, is a structure theorem for plane graphs of bounded path-width. Before stating it, we recall the definition of path-width.

▶ **Definition 3.4.** A *path decomposition* of a graph G is a pair (P, β), where P is a path and β is a function that assigns subsets of $V(G)$, called *bags*, to nodes of P such that

- for each edge $uv \in E(G)$, there exists $x \in V(P)$ such that $\{u, v\} \subseteq \beta(x)$, and
- for every $v \in V(G)$, the set $\{x \in V(P) : v \in \beta(x)\}$ induces a non-empty connected subpath of P.

The *width* of the decomposition is the maximum of $|\beta(x)| - 1$ over all vertices x of P, and the *path-width* of G is the minimum width over all path decompositions of G.

▶ **Theorem 3.5** (*). *Let w, m, and k_0 be non-negative integers, and $g : \mathbf{N} \to \mathbf{N}$ be an arbitrary non-decreasing function. There exist integers w_0 and n_0 such that the following holds. Let G be a 2-connected plane graph and let Y be a set of at most k_0 vertices of G of degree at most 4. If G has path-width at most w and $|V(G)| \geq n_0$, then one of the following holds:*

- *G contains a 0-nest, a 1-nest, a 2-nest, an F-nest, or a degenerate F-nest for some face F of G, of depth m, and with all its cycles or paths disjoint from Y, or*
- *for some $w' \leq w_0$, G contains an (F_1, F_2)-band or a proper (F_1, u)-fan (where F_1 and F_2 are distinct faces and u is a vertex) of length at least $g(w')$ and with support disjoint from Y, such that each of its tiles has size at most w'.*

We pay close attention to explaining Theorem 3.5, because of its great importance in this paper. Comparing it to Definition 3.4, one may think that there is not much difference – the bags $\beta(x)$ of a path decomposition of G of width at most w' might perhaps play the role of tiles of the band or fan in the second conclusion. Unfortunately, this simple idea is quite far from the truth. The subgraphs induced by the bags may not be "drawn locally", that is, its edges may be geometrically far apart in the plane graph G. As an example, consider the width 2 path decomposition of a cycle where one of the vertices of the cycle appears in all the bags.

The main message of Theorem 3.5 thus is that in a plane graph of bounded path-width we can find a long band which is "drawn locally" and decomposes into well-defined small and *connected* tiles (cf. Definition 3.2). Otherwise, such a graph must contain some kind of a deep nest or fan. However, as we will see soon in Corollary 3.6, the latter structures are impossible in the planarizations of optimal drawings of crossing-critical graphs.

The proof of Theorem 3.5 requires some preparatory work, and it uses tools of structural graph theory and of semigroup theory in algebra. Since these tools are quite far from the main topic of this paper, we defer their presentation and an outline of their application towards Theorem 3.5 till Section 5. Instead, we now continue with an application of the theorem in the study of crossing-critical graph structure, as a strengthening of Theorem 2.4.

▶ **Corollary 3.6.** *Let c be a positive integer, and let $g : \mathbf{N} \to \mathbf{N}$ be an arbitrary non-decreasing function. There exist integers w_0 and n_0 such that the following holds. Let G be a 2-connected c-crossing-critical graph, and let G' be the plane graph associated with a drawing of G with the minimum number of crossings. Let Y denote the set of crossing vertices of G'. If $|V(G)| \geq n_0$, then for some $w' \leq w_0$, G' contains an (F_1, F_2)-band or a proper (F_1, u)-fan (where F_1 and F_2 are distinct faces and u is a vertex) of length at least $g(w')$ and with support disjoint from Y, such that each of its tiles has size at most w'.*

Proof. Let $k_0 = \lceil 5c/2 + 16 \rceil$, $w = \lceil 2^{6(72 \log_2 c + 248)c^3 + 1} \rceil + k_0$ and $m = 15c^2 + 105c + 16$. Let w_0 and n_0 be the corresponding integers from Theorem 3.5.

By Theorem 2.3, each c-crossing-critical graph has a drawing with at most k_0 crossings, and thus $|Y| \leq k_0$. By Theorem 2.4, G has path-width at most $w - k_0$, and thus G' has path-width at most w. Hliněný and Salazar [8] and Hernandez-Velez et al. [6] proved that the graph G' obtained from a c-crossing-critical graph G as described does not contain a 0-, 1- and 2-nests of depth m with cycles disjoint from Y. Furthermore, arguments analogous to (some of) those used in [7] can prove that no face F of G' has an F-nest or a degenerate F-nest of depth m with paths disjoint from Y. Further details are left for the full paper. ◀

4 Removing and inserting tiles

In the second part of the paper, we study an arrangement of bounded tiles in a long enough plane band or fan (as described by Corollary 3.6), focusing on finding repeated subsequences which then could be shortened. Importantly, this shortening preserves c-crossing-criticality. In the opposite direction we then manage to define the converse operation of "expansion" of a plane band which also preserves c-crossing-criticality. These findings will imply the final outcome – a construction of all c-crossing-critical graphs from an implicit list of base graphs of bounded size. The formal statement can be found in Theorem 4.9.

Again, we start with a few relevant technical terms. Recall Definition 3.1.

▶ **Definition 4.1** (subband, necklace and shelled band)**.** Let $\mathcal{P} = (P_1, \ldots, P_m)$ be an (F_1, F_2)-band or an (F_1, u)-fan in a 2-connected plane graph. A *subband* or *subfan* consists of a contiguous subinterval $(P_i, P_{i+1}, \ldots, P_j)$ of the band or fan (and its *support* is a subset of the support of the original band or fan).

We say that the band $\mathcal{P}$ is a *necklace* if each of its paths consists of exactly one vertex. A tile (cf. Definition 3.2) of the band or fan $\mathcal{P}$ is *shelled* if it is bounded by a cycle, consisting of two consecutive paths P_i and P_{i+1} of $\mathcal{P}$ and parts of the boundary of F_1 and F_2 (respectively, u), and the two paths P_i, P_{i+1} delimiting the tile have at least two vertices each. The band or fan $\mathcal{P}$ is *shelled* if each of its tiles is shelled. See Figure 4.

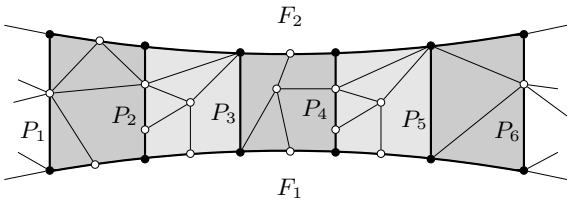


Figure 4 An example of an (F_1, F_2)-band of length 6; this band is shelled (cf. Definition 4.1) and the bounding cycles of the tiles are emphasized in bold lines.

One can easily show that, regarding the outcome of Corollary 3.6, there are only the following two refined subcases that have to be considered in further analysis:

▶ **Lemma 4.2 (*).** *Let w be a positive integer and $f : \mathbf{N} \to \mathbf{N}$ be an arbitrary non-decreasing function. There exist integers n_0 and w' such that the following holds. Let G be a 2-connected plane graph, and let $\mathcal{P} = (P_1, \ldots, P_m)$ be an (F_1, F_2)-band or a proper (F_1, u)-fan in G of length $m \geq n_0$, with all tiles of size at most w. Then either G contains a shelled subband or subfan of $\mathcal{P}$ of length $f(w)$, or G contains a necklace of length $f(w')$ with tiles of size at most w' whose support is contained in the support of $\mathcal{P}$.*

Reducing a necklace. Among the two subcases left by Lemma 4.2, the easier one is that of a necklace which can be reduced simply to a bunch of parallel edges; see also Figure 5.

▶ **Lemma 4.3.** *Let c be a non-negative integer. Let G be a 2-connected c-crossing-critical graph, and let G' be the planarization of a drawing of G with the smallest number of crossings. Let Y denote the set of crossing vertices of G'. Suppose that $\mathcal{P} = (v_1, \ldots, v_m)$, where $m \geq 2$, is a necklace in G' whose support is disjoint from Y. Then for some $p \leq c$, the support of $\mathcal{P}$ consists of p pairwise edge-disjoint paths from v_1 to v_m. Furthermore, the graph G_0 obtained from G by removing the support of $\mathcal{P}$ except for v_1 and v_m and by adding p parallel edges between v_1 and v_m is c-crossing-critical.*

Proof. Let G_1 denote the subgraph of G obtained by removing the support of $\mathcal{P}$ except for v_1 and v_m. Let p be the maximum number of pairwise edge-disjoint paths from v_1 to v_m in the support S of $\mathcal{P}$. Suppose for a contradiction that either $p \geq c+1$ or some edge e of S is not contained in an edge-cut of size p separating v_1 from v_m. In the former case, let e be an arbitrary edge of S. Let $q = c$ if $p \geq c+1$ and $q = p$ otherwise.

By criticality of G, the graph $G - e$ can be drawn in the plane with at most $c-1$ crossings. Consider the drawing of G_1 induced by this drawing, and let a be the minimum number of edges that have to be crossed by any curve in the plane from v_1 to v_m and otherwise disjoint from $V(G_1)$. Note that $a \geq 1$, since otherwise we could draw S without crossings between v_1 and v_m, obtaining a drawing of G with fewer than c crossings. Since $G - e$ contains q pairwise edge-disjoint paths from v_1 to v_m which are not contained in G_1, we conclude that $\mathrm{cr}(G-e) \geq \mathrm{cr}(G_1) + aq \geq q$. Since $\mathrm{cr}(G-e) < c$, we have $q < c$. It follows that $q = p$ and $\mathrm{cr}(G_1) < c - ap$. However, S contains an edge-cut C of order p separating v_1 from v_m by Menger's theorem, and we can add S to the drawing G_1 so that exactly the edges of C are crossed, and each of them exactly a times (by drawing the part of S between v_1 and C close to v_1, and the part of S between v_m and C close to v_m). This way, we obtain a drawing of G with $\mathrm{cr}(G_1) + ap < c$ crossings. This is a contradiction, which shows that $p \leq c$ and that S is the union of p edge-disjoint paths from v_1 to v_m.

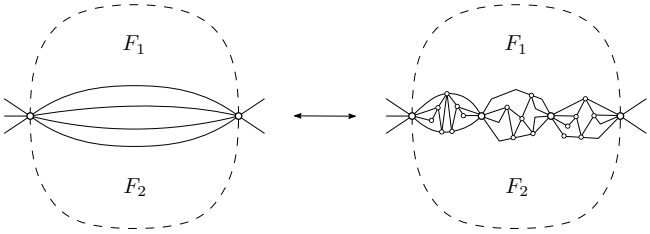


Figure 5 Inserting or removing a necklace (cf. Lemma 4.3 with $p = m = 4$).

Any drawing of G_0 can be transformed into a drawing of G with at most as many crossings in the same way as described in the previous paragraph. Thus $\mathrm{cr}(G_0) \geq c$. Consider now any edge e_0 of G_0. If e_0 is one of the parallel edges between v_1 and v_m, then let e' be any edge of S and $p' = p - 1$, otherwise let $e' = e_0$ and $p' = p$. By the c-crossing-criticality of G, there exists a drawing of $G - e'$ with less than c crossings. Consider the induced drawing of $G_1 - e'$, and let a' denote the minimum number of edges in this drawing that have to be crossed by any curve in the plane from v_1 to v_m and otherwise disjoint from $V(G_1)$. Since $S - e'$ contains p' edge-disjoint paths from v_1 to v_m, we conclude that $\mathrm{cr}(G - e') \geq \mathrm{cr}(G_1 - e') + a'p'$. We can add p' edges between v_1 and v_m to the drawing of $G_1 - e'$ to form a drawing of $G_0 - e_0$ with at most $\mathrm{cr}(G_1 - e') + a'p' \leq \mathrm{cr}(G - e') < c$ crossings. Consequently, G_0 is c-crossing-critical. ◀

Observe that replacing a parallel edge of multiplicity p between vertices u and v in a c-crossing-critical graph with any set of p edge-disjoint plane paths from u to v gives another c-crossing-critical graph. So, the reduction of Lemma 4.3 works in the other direction as well. This two-way process is exhibited by an example with $p = m = 4$ in Figure 5.

Reducing a shelled band or fan. If we could follow the same proof scheme as with necklaces also in the remaining cases of shelled bands and fans, then we would already reach the final goal. Unfortunately, the latter cases are more involved, and require some preparatory work. Compared to the easier case of a necklace, the important difference in the case of a shelled band comes from the fact that the band may be drawn not only in the "straight way" but also in the "twisted way" (recall Figure 1). An indication that this is troublesome comes from the result of Hliněný and Derňár [10], who showed that determining the crossing number of a twisted planar tile is NP-complete (and thus it is not determined by a simple parameter such as the number of edge-disjoint paths between its sides). Consequently, the analysis of shelled bands is significantly more complicated than the relatively straightforward proof of Lemma 4.3. The same remark applies to the shelled fans.

That is why we leave the full details and proofs of the remaining cases for the full paper. Before we dive into technical details needed to at least formulate the final result, Theorem 4.9, we present an informal outline of our approach:

1. Having a very long shelled band $\mathcal{P}$ in our graph G, it is easy to see that the isomorphism types of bounded-size tiles in $\mathcal{P}$ must repeat. Moreover, even bounded-length subbands must have isomorphic repetitions. The first idea is to shorten the band between such repeated isomorphic subbands $\mathcal{P}_1$ and $\mathcal{P}_2$ – by identifying the repeated pieces and discarding what was between (cf. Definition 4.5). If the repeated subband is long enough, we can use some rather easy connectivity properties of $\mathcal{P}$ to show that this yields a smaller graph G_1 of crossing number at least c.

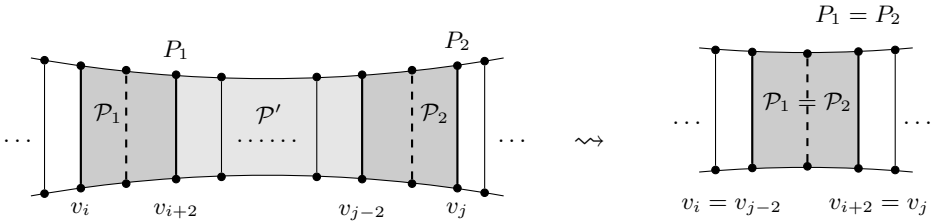


Figure 6 A scheme of a reducible subband $\mathcal{P}'$ (in grey) with repetition $(\mathcal{P}_1, \mathcal{P}_2)$ of order 3 (darker grey), as in Definition 4.5, and the result of the reduction on $\mathcal{P}'$ (on the right).

2. Though, it is not clear that the reduced graph G_1 is c-crossing-critical. Analogously to Lemma 4.3, for any edge $e \in E(G_1)$, we would like to transform a drawing of $G - e$ with less than c crossings to a drawing of $G_1 - e$ with less than c crossings. However, if the drawing of $G - e$ uses some unique properties of the part $\mathcal{P}_{12}$ of the band between $\mathcal{P}_1$ and $\mathcal{P}_2$, we have no way how to mimic this in the drawing of $G_1 - e$ (this is especially troublesome if this part of $G-e$ is drawn in a twisted way, since there is no easy description of what these "unique properties" might be by the NP-completeness result [10]).
   We overcome this difficulty by performing the described reduction only inside longer pieces which repeat elsewhere in the band (cf. Definition 4.6). Hence, in $G_1 - e$ we have many copies of $\mathcal{P}_{12}$, and by appropriate surgery, we can use one of them to mimic the drawing of $\mathcal{P}_{12}$ in $G - e$.
3. A further advantage of reducing within parts that repeat elsewhere is that we can more explicitly describe the converse expansion operation, as duplicating subbands which already exist elsewhere in the (reduced) band.

Let us remark that considering a shelled (F, u)-fan instead of a band is not different, all the arguments simply carry over. The following additional definitions are needed to formalize the outlined claims.

Let $\mathcal{P} = (P_1, \ldots, P_m)$ be an (F_1, F_2)-band or an (F_1, u)-fan in a 2-connected plane graph G, and let T_i be the tile of $\mathcal{P}$ delimited by P_i and P_{i+1}. We say that the band $\mathcal{P}$ is *k-edge-linked* if $k \in \mathbf{N}$ and there exist k pairwise edge-disjoint paths from $V(P_1)$ to $V(P_m)$ contained in the support of $\mathcal{P}$, and for each $i = 1, \ldots, m-1$, the tile T_i contains an edge-cut of size k separating $V(P_i)$ from $V(P_{i+1})$.

Similarly, the fan $\mathcal{P}$ is *k-edge-linked* if there exist k pairwise edge-disjoint paths from $V(P_1)\setminus\{u\}$ to $V(P_m)\setminus\{u\}$ contained in the support of $\mathcal{P}$ minus u, and for each $i = 1, \ldots, m-1$, the sub-tile $T_i - u$ contains an edge-cut of size k separating $V(P_i) \setminus \{u\}$ from $V(P_{i+1}) \setminus \{u\}$. For a closer explanation, one may say that, modulo a trivial adjustment, the fan $\mathcal{P}$ is k-edge-linked iff the corresponding band in $G - u$ is k-edge-linked.

▶ **Definition 4.4** (isomorphic tiles)**.** Two (F_1, F_2)-bands or (F_1, u)-fans $\mathcal{P}_1 = (P_1, \ldots, P_m)$ and $\mathcal{P}_2 = (P'_1, \ldots, P'_m)$ are *isomorphic* if there exists a homeomorphism mapping the support of $\mathcal{P}_1$ to the support of $\mathcal{P}_2$ and mapping the path P_i to P'_i for $i = 1, \ldots, m$, where the paths are taken as directed away from F_1 (i.e., the homeomorphism must map the vertex of P_i incident with F_1 to the vertex of P'_i incident with F_1).

▶ **Definition 4.5** (band or fan reduction)**.** Let G be a graph drawn in the plane with crossings. Let G' be the planarization of G and let Y denote the set of crossing vertices of G'. Let $\mathcal{P}$ be an (F_1, F_2)-band or an (F_1, u)-fan in G' whose support is disjoint from Y. Suppose $\mathcal{P}_1$ and $\mathcal{P}_2$ are isomorphic subbands or subfans of $\mathcal{P}$, with disjoint supports, except for the

vertex u when $\mathcal{P}$ is a fan, and not containing the first and the last path of $\mathcal{P}$. Let $\mathcal{P}'$ be the minimal subband or subfan of $\mathcal{P}$ containing both $\mathcal{P}_1$ and $\mathcal{P}_2$. We then say that $\mathcal{P}'$ is a *reducible subband or subfan with repetition* $(\mathcal{P}_1, \mathcal{P}_2)$. See Figure 6. The *order of* this repetition $(\mathcal{P}_1, \mathcal{P}_2)$ equals the length of $\mathcal{P}_1$ (which is the same as the length of $\mathcal{P}_2$).

Let P_1 and P_2 be the last paths of $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively. Denote by S the support of the subband or subfan between P_1 and P_2, excluding these two paths. Let G_1' be obtained from G' by removing S and by identifying P_1 with P_2 (stretching the drawing of the support of $\mathcal{P}_1$ within the area originally occupied by S). Let G_1 be obtained from G_1' by turning the vertices of Y back into crossings. For clarity, note that the support of $\mathcal{P}'$ is disjoint from Y, and so $\mathcal{P}'$ is also a band or fan in a plane subgraph of G. We then say that G_1 is the *reduction of* G *on* $\mathcal{P}'$.

▶ **Definition 4.6** (t-typical subband or subfan)**.** We say that, in an (F_1, F_2)-band or an (F_1, u)-fan $\mathcal{P}$, a subband $\mathcal{Q}$ is *t-typical* if the following holds: there exist subbands or subfans $\mathcal{P}_1, \ldots, \mathcal{P}_{2t+1}$ of $\mathcal{P}$ appearing in this order, such that they are pairwise isomorphic, with pairwise disjoint supports except for the vertex u when $\mathcal{P}$ is a fan, and $\mathcal{Q} = \mathcal{P}_{t+1}$.

▶ **Lemma 4.7 (*).** *Let G be a 2-connected c-crossing-critical graph drawn in the plane with the minimum number of crossings. Let G' be the planarization of G and let Y denote the set of crossing vertices of G'. Let $c_0 = \lceil 5c/2 + 16 \rceil$ and $k \in \mathbf{N}$. Let $\mathcal{P}$ be a k-edge-linked shelled (F_1, F_2)-band or proper (F_1, u)-fan in G' whose support is disjoint from Y. Let $\mathcal{Q}$ be a subband or subfan of $\mathcal{P}$ which is reducible with repetition of order at least $12c_0 + 2k$. If $\mathcal{Q}$ is c-typical in $\mathcal{P}$, then the reduction G_1 of G on $\mathcal{Q}$ is a c-crossing-critical graph again.*

Expanding a band, fan or a necklace. Finally, it is time to formally define what is a generic converse operation of the instances of reduction considered by Lemmas 4.7 and 4.2:

▶ **Definition 4.8** (n-bounded expansion)**.** Let G be a 2-connected c-crossing-critical graph drawn in the plane with the minimum number of crossings. Let G' be the planarization of G and let Y denote the set of crossing vertices of G'. Let $c_0 = \lceil 5c/2 + 16 \rceil$. Assume $\mathcal{P}$ is a k-edge-linked shelled (F_1, F_2)-band or proper (F_1, u)-fan in G' whose support is disjoint from Y. Let $\mathcal{Q}$ be a c-typical subband or subfan of $\mathcal{P}$ which is reducible with repetition of order at least $12c_0 + 2k$. Let the number of vertices of the support of $\mathcal{Q}$ be at most n, and let G_1 denote the reduction of G on $\mathcal{Q}$. In these circumstances, we say that G is an *n-bounded expansion of* G_1.

Assume $\mathcal{P}'$ is a necklace in G' whose support is disjoint from Y, and let $\mathcal{Q}' = (v_1, v_2)$ be a 1-typical subband of $\mathcal{P}'$ of length 2. Let G_2 be obtained from G by replacing the support S of $\mathcal{Q}'$ by a parallel edge of multiplicity equal to the maximum number of pairwise edge-disjoint paths between v_1 and v_2 in S. Let the number of vertices of the support of $\mathcal{Q}'$ be at most n. In these circumstances, we also say that G is an *n-bounded expansion of* G_1.

▶ **Theorem 4.9 (*).** *For every integer $c \geq 1$, there exists a positive integer n_0 such that the following holds. If G is a 2-connected c-crossing-critical graph, then there exists a sequence $G_0, G_1, \ldots, G_m$ of 2-connected c-crossing-critical graphs such that $|V(G_0)| \leq n_0$, $G_m = G$, and for $i = 1, \ldots, m$, G_i is an n_0-bounded expansion of G_{i-1}.*

Moreover, the generating sequences claimed by Theorem 4.9 can be turned into an efficient enumeration procedure to generate all 2-connected c-crossing-critical graphs of at most given order n, for each fixed c. The output-sensitive complexity of this procedure has polynomial delay in n. We leave further details for the full paper.

5 Deconstructing plane graphs of bounded path-width

We now return to the topic of Section 3, supplementing the technical prerequisites of Theorem 3.5. We need to add a few terms related to Definition 3.4.

Let (P, β) be a path decomposition of a graph G. Let s denote the first node and t the last node of P. For $x \in V(P) \setminus \{s\}$, let $l(x)$ be the node of P preceding x, and let $L(x) = \beta(l(x)) \cap \beta(x)$. For $x \in V(P) \setminus \{t\}$, let $r(x)$ be the node of P following x, and let $R(x) = \beta(r(x)) \cap \beta(x)$. The path decomposition is *proper* if $\beta(x) \not\subseteq \beta(y)$ for all distinct $x, y \in V(P)$. The *interior width* of the decomposition is the maximum over $|\beta(x)| - 1$ over all nodes x of P distinct from s and t. The path decomposition is *p-linked* if $|L(x)| = p$ for all $x \in V(P) \setminus \{s\}$ and G contains p vertex-disjoint paths from $R(s)$ to $L(t)$. The *order* of the decomposition is $|V(P)|$.

A crucial technical step in the proof of Theorem 3.5 is to analyse a topological structure of the bags of a path decomposition (P, β) of a plane graph G, and to find many consecutive subpaths of P on which the decomposition repeats the same "topological behavior". For this we are going to model the bags of the decomposition (P, β) as letters of a string over a suitable finite semigroup (these letters present an abstraction of the bags), and to apply the following algebraic tool, Lemma 5.1.

Let T be a rooted ordered tree (i.e., the order of children of each vertex is fixed). Let f be a function that to each leaf of T assigns a string of length 1, such that for each non-leaf vertex v of T, $f(v)$ is the concatenation of the strings assigned by f to the children of v in order. We say that (T, f) *yields the string* assigned to the root of T by f. If the letters of the string are elements of a semigroup A, then for each $v \in V(T)$, let $f_A(v)$ denote the product of the letters of $f(v)$ in A. Recall that an element e of A is *idempotent* if $e^2 = e$. A tree (T, f) is an *A-factorization tree* if for every vertex v of T with more than two children, there exists an idempotent element $e \in A$ such that $f_A(x) = e$ for each child x of v (and hence also $f_A(v) = e$). Simon [17] showed existence of bounded-depth A-factorization trees for every string; the improved bound in the following lemma was proved by Colcombet [4]:

▶ **Lemma 5.1** ([4]). *For every finite semigroup A and each string of elements of A, there exists an A-factorization tree of depth at most $3|A|$ yielding this string.*

We further need to formally define what we mean by a "topological behavior" of bags and subpaths of a path decomposition of our G. This will be achieved by the following term of a q-type.

In this context we consider multigraphs (i.e., with parallel edges and loops allowed – each loop contributes 2 to degree of the incident vertex, and not necessarily connected) with some of its vertices labelled by distinct unique labels. A plane multigraph G is *irreducible* if G has no faces of size 1 or 2, and every unlabelled vertex of degree at most 2 is an isolated vertex incident with one loop (this loop, hence, cannot bound a 1-face). Two plane multigraphs G_1 and G_2 with some of the vertices labelled are *homeomorphic* if there exists a homeomorphism φ of the plane mapping G_1 onto G_2 so that for each vertex $v \in V(G_1)$, the vertex $\varphi(v)$ is labelled iff v is, and then v and $\varphi(v)$ have the same label. For G with some of its vertices labelled using the labels from a finite set $\mathcal{L}$, the *q-type* of G is the set of all non-homeomorphic irreducible plane multigraphs labelled from $\mathcal{L}$ and with at most q unlabelled vertices, and whose subdivisions are homeomorphic to subgraphs of G.

Let G be a plane graph and let (P, β) be its p-linked path decomposition. Let s and t be the endpoints of P. Fix pairwise vertex-disjoint paths $Q_1, \ldots, Q_p$ between $R(s)$ and $L(t)$. Consider a subpath P' of $P - \{s, t\}$, and let $G_{P'}$ be the subgraph of G induced by $\bigcup_{x \in V(P')} \beta(x)$. If s' and t' are the (left and right) endpoints of P', we define $L(P') = L(s')$

and $R(P') = R(t')$. Let us label the vertices of $G_{P'}$ using (some of) the labels $\{l_1, \ldots, l_p, r_1, \ldots, r_p, c_1, \ldots, c_p\}$ as follows: For $i = 1, \ldots, p$, let u and v be the vertices in which Q_i intersects $L(P')$ and $R(P')$, respectively. If $u \neq v$, we give u the label l_i and v the label r_i. Otherwise, we give $u = v$ the label c_i. For an integer q, the *q-type* of P' is the q-type of $G_{P'}$ with this labelling. If P' contains just one node x, then we speak of the *q-type* of x.

The q-types of subpaths of a linked path decomposition naturally form a semigroup with concatenation of the subpaths, as detailed in the full paper. From Lemma 5.1, specialised to our case, we derive the following structural description which is crucial in the proof of Theorem 3.5. Further technical details are again left for the full paper.

▶ **Theorem 5.2 (*).** *Let w and q be non-negative integers, and let $f : \mathbf{N} \to \mathbf{N}$ be an arbitrary non-decreasing function. There exist integers w_0 and n_0 such that, for any plane graph G that has a proper path decomposition of interior width at most w and order at least n_0, the following holds. For some $w' \leq w_0$ and $p \leq w$, G also has a p-linked proper path decomposition (P, β) of interior width at most w' and order at least $f(w')$, such that for each node x of P distinct from its endpoints, the q-type of x is the same idempotent element.*

In other words, we can find a decomposition in which all topological properties of the drawing that hold in one bag repeat in all the bags. So, for example, if for some node x, the vertices of $L(x)$ are separated in the drawing from vertices of $R(x)$ by a cycle contained in the bag of x, then this holds in every bag, and we conclude that the drawing contains a large 0-nest. Other outcomes of Theorem 3.5 naturally correspond to other possible local properties of the drawings of the bags.

6 Conclusion

To summarize, we have shown a structural characterisation and an enumeration procedure for all 2-connected c-crossing-critical graphs, using bounded-size replication steps over an implicit finite set of base c-crossing-critical graphs. The characterisation can be reused to describe all c-crossing-critical graphs (without the connectivity assumption) since all their proper blocks must be c_i-crossing-critical for some $c_i < c$.

With this characterisation at hand, one can expect significant progress in the crossing number research, both from mathematical and algorithmic perspectives. For example, one can quite easily derive from Theorem 4.9 that, for no c there is an infinite family of 3-regular c-crossing-critical graphs, a claim that has been so far proved only via the Graph minors theorem of Robertson and Seymour. One can similarly expect a progress in some long-time open questions in the area of crossing-critical graphs, such as to improve the bound of Theorem 2.3 or to decide possible existence of an infinite family of 5-regular c-crossing-critical graphs for some c.

References

1. M. Ajtai, V. Chvátal, M.M. Newborn, and E. Szemerédi. Crossing-free subgraphs. In *Theory and Practice of Combinatorics*, volume 60 of *North-Holland Mathematics Studies*, pages 9–12. North-Holland, 1982. doi:10.1016/S0304-0208(08)73484-4.
2. Drago Bokal. Infinite families of crossing-critical graphs with prescribed average degree and crossing number. *Journal of Graph Theory*, 65(2):139–162, 2010. doi:10.1002/jgt.20470.
3. Drago Bokal, Bogdan Oporowski, R. Bruce Richter, and Gelasio Salazar. Characterizing 2-crossing-critical graphs. *Advances in Applied Mathematics*, 74:23–208, 2016. doi:10.1016/j.aam.2015.10.003.

4 Thomas Colcombet. Factorization forests for infinite words and applications to countable scattered linear orderings. *Theor. Comput. Sci.*, 411(4-5):751–764, 2010. doi:10.1016/j.tcs.2009.10.013.

5 Z. Dvořák and B. Mohar. Crossing-critical graphs with large maximum degree. *J. Combin. Theory, Ser. B*, 100:413–417, 2010. doi:10.1016/j.jctb.2009.11.003.

6 C. Hernandez-Velez, G. Salazar, and R. Thomas. Nested cycles in large triangulations and crossing-critical graphs. *Journal of Combinatorial Theory, Series B*, 102:86–92, 2012. doi:10.1016/j.jctb.2011.04.006.

7 P. Hliněný. Crossing-number critical graphs have bounded path-width. *J. Combin. Theory, Ser. B*, 88:347–367, 2003. doi:10.1016/S0095-8956(03)00037-6.

8 P. Hliněný and G. Salazar. Stars and bonds in crossing-critical graphs. *J. Graph Theory*, 65:198–215, 2010. doi:10.1002/jgt.20473.

9 Petr Hliněný. Crossing-critical graphs and path-width. In Petra Mutzel, Michael Jünger, and Sebastian Leipert, editors, *Graph Drawing, 9th International Symposium, GD 2001, Revised Papers*, volume LNCS 2265 of *Lecture Notes in Computer Science*, pages 102–114. Springer, 2002. doi:10.1007/3-540-45848-4_9.

10 Petr Hliněný and Marek Dernár. Crossing number is hard for kernelization. In *32nd International Symposium on Computational Geometry, SoCG 2016, June 14-18, 2016, Boston, MA, USA*, volume 51 of *LIPIcs*, pages 42:1–42:10. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPIcs.SoCG.2016.42.

11 Martin Kochol. Construction of crossing-critical graphs. *Discrete Mathematics*, 66(3):311–313, 1987. doi:10.1016/0012-365X(87)90108-7.

12 Tom Leighton. *Complexity Issues in VLSI.* Foundations of Computing Series. MIT Press, Cambridge, MA, 1983.

13 B. Mohar and C. Thomassen. *Graphs on Surfaces.* The Johns Hopkins University Press, Baltimore and London, 2001.

14 Benny Pinontoan and R. Bruce Richter. Crossing numbers of sequences of graphs II: Planar tiles. *Journal of Graph Theory*, 42(4):332–341, 2003. doi:10.1002/jgt.10097.

15 R. Bruce Richter and Carsten Thomassen. Minimal graphs with crossing number at least *k*. *J. Comb. Theory, Ser. B*, 58(2):217–224, 1993. doi:10.1006/jctb.1993.1038.

16 Gelasio Salazar. Infinite families of crossing-critical graphs with given average degree. *Discrete Mathematics*, 271(1-3):343–350, 2003. doi:10.1016/S0012-365X(03)00136-5.

17 Imre Simon. Factorization forests of finite height. *Theor. Comput. Sci.*, 72(1):65–94, 1990. doi:10.1016/0304-3975(90)90047-L.

18 László A. Székely. Crossing numbers and hard Erdös problems in discrete geometry. *Combinatorics, Probability & Computing*, 6(3):353–358, 1997. URL: http://journals.cambridge.org/action/displayAbstract?aid=46513.

19 Jozef Širáň. Infinite families of crossing-critical graphs with a given crossing number. *Discrete Mathematics*, 48(1):129–132, 1984. doi:10.1016/0012-365X(84)90140-7.

The Multi-cover Persistence of Euclidean Balls

Herbert Edelsbrunner
IST Austria (Institute of Science and Technology Austria)
Am Campus 1, 3400 Klosterneuburg, Austria
edels@ist.ac.at
https://orcid.org/0000-0002-9823-6833

Georg Osang
IST Austria (Institute of Science and Technology Austria)
Am Campus 1, 3400 Klosterneuburg, Austria
georg.osang@ist.ac.at

Abstract

Given a locally finite $X \subseteq \mathbb{R}^d$ and a radius $r \geq 0$, the *k-fold cover* of X and r consists of all points in $\mathbb{R}^d$ that have k or more points of X within distance r. We consider two filtrations – one in *scale* obtained by fixing k and increasing r, and the other in *depth* obtained by fixing r and decreasing k – and we compute the persistence diagrams of both. While standard methods suffice for the filtration in scale, we need novel geometric and topological concepts for the filtration in depth. In particular, we introduce a rhomboid tiling in $\mathbb{R}^{d+1}$ whose horizontal integer slices are the order-k Delaunay mosaics of X, and construct a zigzag module from Delaunay mosaics that is isomorphic to the persistence module of the multi-covers.

2012 ACM Subject Classification Mathematics of computing → Combinatorial algorithms

Keywords and phrases Delaunay mosaics, hyperplane arrangements, discrete Morse theory, zigzag modules, persistent homology

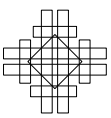
Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.34

Funding This work is partially supported by the DFG Collaborative Research Center TRR 109, 'Discretization in Geometry and Dynamics', through grant no. I02979-N35 of the Austrian Science Fund (FWF).

1 Introduction

The work in this paper is motivated by density fluctuations in point configurations. These fluctuations can be large – and the task may be the identification of regions with a prescribed density profile – or they can be small – and the goal may be to pick up subtle variations. For example, we may want to quantify local defects in lattice configurations or describe long-range differences between similar configurations, such as the *face-centered cubic* (FCC) lattice and the *hexagonal close-packed* (HCP) configuration in $\mathbb{R}^3$. While both give densest sphere packings in $\mathbb{R}^3$, physical particle systems prefer to settle in the FCC configuration. The reason for this preference is not well understood. Our quantification of the long-range effects of density differences discriminates between the two configurations and this way sheds light on this phenomenon.

Using standard methods from computational geometry and topology, we describe mathematical and computational tools to quantify density fluctuations. Our work is closely related to the *distance to a measure* introduced in [6]. As demonstrated in a follow-up paper [14], this distance can be approximated using the *order-k Voronoi tessellation* of the configuration,




a concept introduced in the early days of computational geometry [20], but see also [11, 16]. Order-k Voronoi tessellations are also at the core of our work:

1. Given a locally finite set $X \subseteq \mathbb{R}^d$, we introduce a rhomboid tiling in $\mathbb{R}^{d+1}$ whose horizontal slices at integer depths are the geometric duals of the order-k Voronoi tessellations.

We call these duals the *order-k Delaunay mosaics* of X. The tiling clarifies the structure of individual mosaics and the relationship between them. Restricting the order-k Voronoi tessellation to the *k-fold cover* of the balls with radius $r \geq 0$ centered at the points in X, we get a subcomplex of the order-k Delaunay mosaic; see [15] for the introduction of this concept for statistical purposes in two dimensions. Our second result makes use of the family of such subcomplexes obtained by varying the scale:

2. Fixing k and varying r, we compute the persistence diagram of the density fluctuations from the filtration of order-k Delaunay mosaics of X.

The ingredients for our second result are standard, but to get the actual results, we needed an implementation of the order-k Delaunay mosaic algorithm, which we developed based on the rhomboid tiling. In contrast to [2, 19], this gives a simple implementation, which we will describe elsewhere. Using this software in $\mathbb{R}^3$, we find that the FCC and the HCP configurations have the same persistence diagram for $k = 1, 2, 3$ but different persistence diagrams for $k = 4, 5$. Our third result is an algorithm for the persistence of the multi-covers obtained by varying the depth:

3. Fixing r and varying k, we compute the persistence diagram of the filtration of multi-covers from the rhomboid tiling of X.

Several innovative adaptations of the standard approach to persistence are needed to get our third result. The main challenge is the combinatorial difference of the Delaunay mosaics from one value of k to the next. Here we use the rhomboid tiling to establish a zigzag module whose persistence diagram is the same as that of the filtration of multi-covers. We get the persistence diagram using the algorithm in [4, 5]. Our work is also related to the study of multi-covers based on Čech complexes in [21]. While the relation between the different Čech complexes is simpler than that between the Delaunay mosaics, their explosive growth for increasing radius leads to algorithms with prohibitively long running time.

Outline. Section 2 describes the rhomboid tiling in $\mathbb{R}^{d+1}$ that encodes the Delaunay mosaics of all orders of a locally finite set in $\mathbb{R}^d$. Section 3 relates the k-fold covers with the order-k Delaunay mosaics and introduces radius functions on the rhomboid tiling and the mosaics. Section 4 introduces slices of a tiling at half-integer depths and explains how they are used to compute the persistence diagram in depth. Section 5 concludes the paper.

2 Rhomboid tiling

Given a locally finite set in $\mathbb{R}^d$, we are interested in the collection of Delaunay mosaics of all orders. Assuming the set is in general position, there exists a rhomboid tiling in $\mathbb{R}^{d+1}$ such that the Delaunay mosaics are horizontal slices of the tiling. This section introduces the tiling and proves the relation to Delaunay mosaics.

Rhomboid tiling. Let $X \subseteq \mathbb{R}^d$ be locally finite and in general position. Every $(d-1)$-dimensional sphere, S, in $\mathbb{R}^d$ partitions X into the points *inside*, *on*, and *outside* S. We call this the *ordered three-partition* of X defined by S, and denote it as $X = \text{In}(S) \cup \text{On}(S) \cup \text{Out}(S)$.

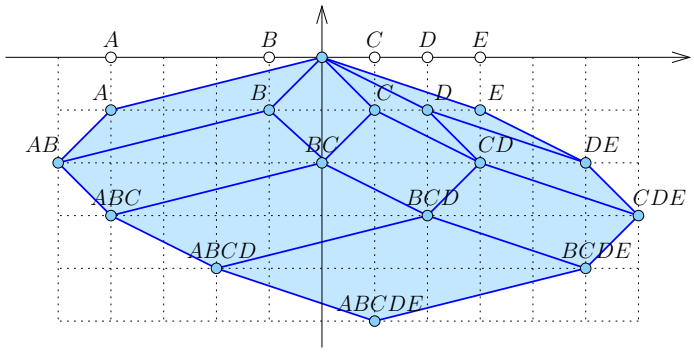


Figure 1 The rhomboid tiling of 5 points on the real line. For example, the upper left 2-dimensional rhomboid defined by $(\emptyset, \{A, B\}, \{C, D, E\})$ is the convex hull of the points $y_\emptyset$, y_A, y_B, and $y_{\{A,B\}}$. The horizontal line at depth k intersects the tiling in a geometric realization of the order-k Delaunay mosaic of the 5 points.

By assumption of general position, we have $0 \leq |\mathrm{On}(S)| \leq d+1$, but there are no a priori upper bounds on the sizes of the other two sets.

We map each ordered three-partition defined by a $(d-1)$-sphere, S, to a parallelepiped in $\mathbb{R}^{d+1}$, which we call the *rhomboid* of S, denoted $\mathrm{rho}(S)$. To define it, we write $y_x = (x, -1) \in \mathbb{R}^{d+1}$, for every $x \in X$, and $y_Q = \sum_{x \in Q} y_x$ for every $Q \subseteq X$. The $(d+1)$-st coordinate of y_Q is therefore $-|Q|$, and we call $|Q|$ the *depth* of the point. With this notation, $\mathrm{rho}(S) = \mathrm{conv}\,\{y_Q \mid \mathrm{In}(S) \subseteq Q \subseteq \mathrm{In}(S) \cup \mathrm{On}(S)\}$. Equivalently, $\mathrm{rho}(S)$ is the rhomboid spanned by the vectors y_x, with $x \in \mathrm{On}(S)$, and translated along $y_{\mathrm{In}(S)}$. Its dimension is the number of spanning vectors, $|\mathrm{On}(S)|$. Observe that every face of $\mathrm{rho}(S)$ is again the rhomboid defined by a sphere. To see this, we note that for every ordered partition of the points on S into three sets, $\mathrm{On}(S) = O_{in} \cup O_{on} \cup O_{out}$, there is a sphere S' with $\mathrm{In}(S') = \mathrm{In}(S) \cup O_{in}$, $\mathrm{On}(S') = O_{on}$, and $\mathrm{Out}(S') = \mathrm{Out}(S) \cup O_{out}$. There are $3^{|\mathrm{On}(S)|}$ such ordered partitions, and each corresponds to a face of $\mathrm{rho}(S)$. By definition, the *rhomboid tiling* of X, denoted $\mathrm{Rho}(X)$, is the collection of all rhomboids defined by spheres; see Figure 1. As suggested by the figure, the ordered three partition $(\emptyset, \emptyset, X)$ is mapped to the origin of $\mathbb{R}^{d+1}$. We claim the following properties.

▶ **Theorem 1** (Rhomboid Tiling). *Let $X \subseteq \mathbb{R}^d$ be locally finite and in general position. Then*

1. $\mathrm{Rho}(X)$ *is dual to an arrangement of hyperplanes in* $\mathbb{R}^{d+1}$*;*
2. $\mathrm{Rho}(X)$ *is the projection of the boundary of a zonotope in* $\mathbb{R}^{d+2}$*;*
3. *the horizontal slice of* $\mathrm{Rho}(X)$ *at depth k is the order-k Delaunay mosaic of X.*

Note that Claim 2 in Theorem 1 implies that the rhomboid tiling is a geometric realization of the dual of the arrangement in $\mathbb{R}^{d+1}$, that is: its rhomboids intersect in common faces but not otherwise. The remainder of this section proves the three claims. To keep the proofs self-contained, we will define hyperplane arrangements and order-k Delaunay mosaics before we use them. We refer to [7] for additional information on their relation to point configurations.

Proof of Claim 1: hyperplane arrangement. For each point $x \in X$, write $f_x \colon \mathbb{R}^d \to \mathbb{R}$ for the affine map defined by $f_x(p) = \langle p, x \rangle - \|x\|^2/2 = (\|p\|^2 - \|p - x\|^2)/2$. The graph of f_x is a hyperplane in $\mathbb{R}^{d+1}$ that is tangent to the paraboloid consisting of the points $(p, z) \in \mathbb{R}^d \times \mathbb{R}$ that satisfy $z = \|p\|^2/2$. The collection of such hyperplanes decomposes $\mathbb{R}^{d+1}$ into convex cells, which we call the *hyperplane arrangement* of X, denoted $\mathrm{Arr}(X)$; see Figure 2. The *cells*

in the arrangement are intersections of hyperplanes and closed half-spaces. More formally, for each cell there is an ordered three-partition $X = X_{in} \cup X_{on} \cup X_{out}$ such that the cell consists of all points $(p, z) \in \mathbb{R}^d \times \mathbb{R}$ that satisfy

$$z \leq f_x(p) \quad \text{if } x \in X_{in}, \tag{1}$$
$$z = f_x(p) \quad \text{if } x \in X_{on}, \tag{2}$$
$$z \geq f_x(p) \quad \text{if } x \in X_{out}. \tag{3}$$

Since X is assumed to be in general position, the dimension of the cell is $i = d + 1 - |X_{on}|$. Turning the non-strict into strict inequalities, we get the interiors of the cells, which partition $\mathbb{R}^{d+1}$. We refer to the i-dimensional cells as *i-cells* and to the $(d+1)$-cells as *chambers*.

Importantly, there is a bijection between the cells of $\mathrm{Arr}(X)$ and the rhomboids in $\mathrm{Rho}(X)$. To see this, map a point (p, z) in the interior of a cell to the sphere S with center p and squared radius $r^2 = \max\{0, \|p\|^2 - 2z\}$. Using the definition of f_x, we observe that $\mathrm{In}(S) = X_{in}$, $\mathrm{On}(S) = X_{on}$, and $\mathrm{Out}(S) = X_{out}$. We can reverse the map, and while this will not reach the points with $\|p\|^2 - 2z < 0$, these points all belong to the chamber of the ordered three-partition $(\emptyset, \emptyset, X)$. This establishes the bijection between the cells and the rhomboids. This bijection reverses dimensions and preserves incidences, which justifies that we call it a *duality* between the rhomboid tiling and the hyperplane arrangement. This completes the proof of Claim 1 in Theorem 1. ◀

Proof of Claim 2: zonotope. We recall that a *zonotope* is a special convex polyhedron, namely one obtained by taking the Minkowski sum of finitely many line segments. The zonotope of interest is constructed from the line segments that connect the origin to the points $v_x = (x, -1, \|x\|^2/2) \in \mathbb{R}^{d+2}$, with $x \in X$. Note that these line segments project to the vectors $y_x = (x, -1)$ used to build the rhomboid tiling. By construction, y_x is normal to the graph of f_x, which is the zero set of $F_x \colon \mathbb{R}^{d+1} \to \mathbb{R}$ defined by $F_x(q) = \langle q, y_x \rangle - \|x\|^2/2$; see Figure 2. Adding a $(d+2)$-nd coordinate, w, we introduce $G_x \colon \mathbb{R}^{d+2} \to \mathbb{R}$ defined by $G_x(q, w) = \langle q, y_x \rangle + w\|x\|^2/2$. Its zero-set is normal to v_x, the restriction of $G_x^{-1}(0)$ to $w = -1$ is the zero-set of F_x, and $G_x(0) = 0$. In other words, if we identify $\mathbb{R}^{d+1}$ with the hyperplane $w = -1$ in $\mathbb{R}^{d+2}$, then the zero-sets of the G_x intersect $\mathbb{R}^{d+1}$ in $\mathrm{Arr}(X)$ and they all pass through the origin in $\mathbb{R}^{d+2}$.

By construction, the thus defined zonotope is dual to the arrangement of hyperplanes $G_x^{-1}(0)$ for $x \in X$. Therefore, the antipodal face pairs of the zonotope correspond dually to the cells of $\mathrm{Arr}(X)$, provided we interpret the arrangement projectively, which means we combine antipodal pairs of unbounded cells; see also [7, Section 1.7]. We get a more direct dual correspondence by projecting the bottom side of the boundary of the zonotope to $\mathbb{R}^{d+1}$. By choice of the line segments, the vertices on this side project vertically to the vertices of $\mathrm{Rho}(X)$, and since both are dual to $\mathrm{Arr}(X)$, we conclude that $\mathrm{Rho}(X)$ is the projection of this side of the zonotope. This completes the proof of Claim 2 in Theorem 1. ◀

Proof of Claim 3: Delaunay mosaics. We begin with some definitions. The *Voronoi domain* of $Q \subseteq X$ is $\mathrm{dom}(Q) = \{p \in \mathbb{R}^d \mid \|p - x\| \leq \|p - y\|, \forall x \in Q, \forall y \in X \setminus Q\}$. Its *order* is $|Q|$. For each Voronoi domain, there is a chamber in $\mathrm{Arr}(X)$ that projects vertically to the domain. Indeed, the chamber is defined by the ordered three-partition $X = X_{in} \cup X_{on} \cup X_{out}$ with $X_{in} = Q$, $X_{on} = \emptyset$, and $X_{out} = X \setminus Q$. For each positive integer k, the *order-k Voronoi tessellation* is $\mathrm{Vor}_k(X) = \{\mathrm{dom}(Q) \mid |Q| = k\}$. We can construct it by projecting all chambers whose ordered three-partitions satisfy $|X_{in}| = k$ and $|X_{on}| = 0$; see [7, Chapter 13] or [10]. These chambers correspond to the vertices of the rhomboid tiling at depth k.

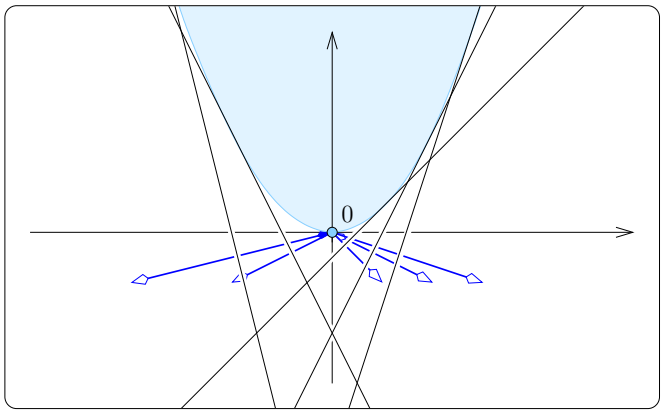


Figure 2 A portion of the arrangement formed by the lines (hyperplanes) that are the graphs of the f_x, with $x \in X$. These lines are tangent to the paraboloid and normal to the vectors $y_x = (x, -1)$. The topmost chamber contains the paraboloid.

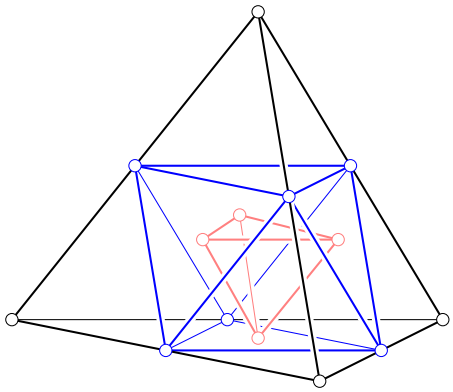


Figure 3 The convex hulls of the barycenters of the $(j-1)$-faces of the tetrahedron. From *outside in*: the tetrahedron for $j = 1$, the octahedron for $j = 2$, and another tetrahedron for $j = 3$.

Since $\mathrm{Rho}(X)$ is dual to $\mathrm{Arr}(X)$, we get the dual of the Voronoi tessellation by taking the slice $z = -k$ of $\mathrm{Rho}(X)$. However the dual of the order-k Voronoi tessellation is precisely the order-k Delaunay mosaic [2]. This completes the proof of Claim 3 in Theorem 1. ◀

We see that the cells of $\mathrm{Del}_k(X)$ are special slices of the rhomboids. Combinatorially, they are equivalent to slices of the unit cube that are orthogonal to the main diagonal and pass through non-empty subsets of the vertices. For the $(d+1)$-cube, there are $d+2$ such slices, which we index from 0 to $d+1$. The j-th slice passes through $\binom{d+1}{j}$ vertices, so we have a vertex for $j = 0, d+1$ and a d-simplex for $j = 1, d$. To describe these slices in general, let U_{d+1} be the $d+1$ unit coordinate vectors. The j-th slice is the convex hull of the points $\sum_{u \in Q} u$ with $Q \in \binom{U_{d+1}}{j}$, in which the empty sum is $(0, 0) \in \mathbb{R}^d \times \mathbb{R}$, by convention. To get an intuition, it might be easier to divide the sums by j, in which case the j-th slice is the convex hull of the barycenters of the $(j-1)$-faces of the standard d-simplex; see Figure 3.

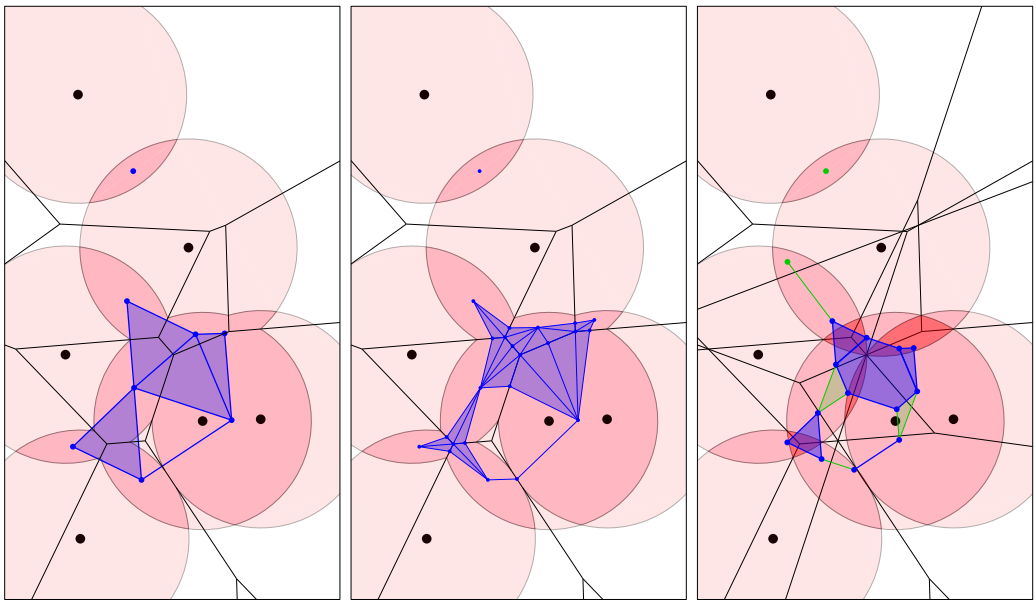


Figure 4 *Left panel*: six points in the plane and a pink ball of radius r centered at each. The black order-2 Voronoi tessellation decomposes the 2-fold cover into convex domains. The corresponding subcomplex of the dual order-2 Delaunay mosaic is superimposed in blue. *Middle panel*: the barycentric subdivision of the order-2 Delaunay mosaic and its geometric realization inside the 2-fold cover. *Right panel*: the black order-2.5 Voronoi tessellation decomposes the 2- and 3-fold covers into convex domains each. The dual complexes are $D_{2.5}$, whose cells are blue, and $E_{2.5}$, whose additional cells are green.

3 Multi-covers

In this section, we exploit the rhomboid tiling to shed light on the filtration of multi-covers we get by varying the radius. The main new insight is that the discrete function on the Delaunay mosaic that encodes this filtration is a relaxation of a standard discrete Morse function. We begin with a formal introduction of the multi-covers.

***k*-fold cover.** Let $X \subseteq \mathbb{R}^d$ be locally finite. Given a radius $r \geq 0$, the *k-fold cover* of X and r consists of all points $p \in \mathbb{R}^d$ for which there are k or more points $x \in X$ with $\|x - p\| \leq r$, or in other words, the points $p \in \mathbb{R}^d$ that are covered by at least k of the balls of radius r around the points $x \in X$. Denoting this set by $\text{Cover}_k(X, r)$, we have

$$\text{Cover}_k(X, r) \subseteq \text{Cover}_k(X, s), \tag{4}$$

$$\text{Cover}_k(X, r) \subseteq \text{Cover}_\ell(X, r), \tag{5}$$

whenever $r \leq s$ and $\ell \leq k$. We are interested in computing the persistent homology of the multi-covers, both in the direction of increasing radius and in the direction of decreasing order. To do so, we represent the covers by complexes, namely by subcomplexes of the Delaunay mosaics. Varying the radius, we get a nested sequence of subcomplexes of the order-k Delaunay mosaic, and the persistent homology can be computed with standard methods; see e.g. [8, Chapter VII]. Varying the order, on the other hand, we get subcomplexes of different Delaunay mosaics, and we need a novel algorithm to compute persistent homology.

Before we discuss this algorithm in Section 4, we note that the order-k Voronoi tessellation decomposes the k-fold cover into convex sets. To see this, let $|Q| = k$ and define $\text{dom}(Q, r) =$

$\mathrm{dom}(Q) \cap \mathrm{Cover}_k(Q,r)$, which is an intersection of convex sets and therefore convex. We write $\mathrm{Vor}_k(X,r)$ for the collection of domains $\mathrm{dom}(Q,r)$ with $|Q| = k$, and since $\mathrm{dom}(Q,r) = \mathrm{dom}(Q) \cap \mathrm{Cover}_k(X,r)$, we refer to this as the *Voronoi decomposition* of $\mathrm{Cover}_k(X,r)$. Since $\mathrm{dom}(Q,r) \subseteq \mathrm{dom}(Q)$, the dual of this decomposition is a subcomplex of the order-k Delaunay mosaic, which we denote $\mathrm{Del}_k(X,r) \subseteq \mathrm{Del}_k(X)$. See the left panel in Figure 4 for an example. Modulo a technicality caused by the mosaic not necessarily being simplicial, the Nerve Theorem [17] implies that the cover and the mosaic have the same homotopy type. We state this fact more formally and without proof.

▶ **Lemma 2** (Almost Nerve). *Let $X \subseteq \mathbb{R}^d$ be locally finite and in general position. For every integer $k \geq 1$ and real $r \geq 0$, $\mathrm{Del}_k(X,r)$ and $\mathrm{Cover}_k(X,r)$ have the same homotopy type.*

The radius function on the rhomboid tiling. To shed additional light on the subcomplexes of the Delaunay mosaics, we introduce a discrete function on the collection of rhomboids discussed in Section 2. Calling it the *radius function*, $\mathcal{R}\colon \mathrm{Rho}(X) \to \mathbb{R}$, we define it by remembering that each j-dimensional rhomboid, $\rho \in \mathrm{Rho}(X)$, corresponds to a $(d+1-j)$-dimensional cell, $\rho^* \in \mathrm{Arr}(X)$. Decomposing a point of the cell into its first d coordinates and its $(d+1)$-st coordinate, we write $q = (p,z) \in \mathbb{R}^d \times \mathbb{R}$, and we define $r(q) = \|p\|^2 - 2z$. With this notation, we define the radius function by mapping ρ to the minimum value of any point in its dual cell:

$$\mathcal{R}(\rho) = \min_{q \in \rho^*} r(q). \tag{6}$$

By convention, the value of the vertex that corresponds to the ordered three-partition $X = (\emptyset, \emptyset, X)$ is $\mathcal{R}(0) = -\infty$. To obtain a geometric interpretation of this construction, consider the paraboloid defined by the equation $z = \frac{1}{2}\|p\|^2$ in $\mathbb{R}^{d+1}$ and introduce $\pi_t(p)\colon \mathbb{R}^d \to \mathbb{R}$ defined by $\pi_t(p) = \frac{1}{2}(\|p\|^2 - t)$. The image of π_t is the original paraboloid dropped vertically down by a distance $\frac{t}{2}$. With this notation, $\mathcal{R}(\rho)$ is the minimum t such that the image of π_t has a non-empty intersection with ρ^*.

Clearly, $\mathcal{R}$ is *monotonic*, that is: $\mathcal{R}(\rho) \leq \mathcal{R}(\varrho)$ if ρ is a face of ϱ. Indeed, if ρ is a face of ϱ, then ϱ^* is a face of ρ^*, which implies that the paraboloid touches ρ^* at the same time or before it touches ϱ^* when dropped. It follows that the sublevel sets of the radius function are subcomplexes of the rhomboid tiling. For X in general position, the radius function satisfies the stronger requirement of a generalized discrete Morse function; see [12, 13]. To explain what this means, let $f\colon \mathrm{Rho}(X) \to \mathbb{R}$ and for each $r \in \mathbb{R}$ consider the *Hasse diagram*, defined as the graph whose nodes are the rhomboids in $f^{-1}(r)$, with an arc connecting two nodes if one rhomboid is a face of the other. The *steps* of f are the components of the graphs representing the level sets of f. Note that the steps partition $\mathrm{Rho}(X)$. We call f a *generalized discrete Morse function* if each step is an *interval*, meaning there are rhomboids $\rho \subseteq \varrho$ such that the step consists of all rhomboids that are faces of ϱ and contain ρ as a face. It is useful to distinguish between *singular intervals*, when $\rho = \varrho$, and *non-singular intervals*, when ρ is a proper face of ϱ. Indeed, consider two contiguous sublevel sets that differ by a level set: $f^{-1}[-\infty, r] \setminus f^{-1}[-\infty, r) = f^{-1}(r)$. If this difference is a non-singular interval, then the two sublevel sets have the same homotopy type, while if the difference is a singular interval, then they have different homotopy types. We prove that the radius function is a generalized discrete Morse function with the additional property that every sublevel set is contractible.

▶ **Lemma 3** (Generalized Discrete Morse). *Let $X \subseteq \mathbb{R}^d$ be locally finite and in general position. Then $\mathcal{R}\colon \mathrm{Rho}(X) \to \mathbb{R}$ is a generalized discrete Morse function. Furthermore, all intervals in the implied partition have a vertex as a lower bound, and there is only one singular interval, which contains the vertex at the origin.*

Proof. The vertex at the origin corresponds to the three-partition $(\emptyset, \emptyset, X)$, has radius $\mathcal{R}(0) = -\infty$, and forms a singular interval. Every other interval is defined by a point $q \in \mathbb{R}^{d+1}$ at which the dropping paraboloid first touches a cell of the arrangement. There is one such point on every plane that is the common intersection of hyperplanes forming the arrangement. By general position, all these points are different. Let q belong to an i-plane, which is common to $j = d+1-i$ hyperplanes. It belongs to the interior of an i-cell, which is common to 2^j chambers. Exactly one of these chambers has not already been touched before the i-cell. The paraboloid touches this chamber at the same point q and similarly every cell that is a face of this chamber and contains the i-cell as a face. The corresponding rhomboids form an interval of the radius function, with an upper bound of dimension j and a lower bound of dimension 0. We have $1 \leq j \leq d+1$, which implies that the interval is not singular.

To show that $\mathcal{R}$ is a generalized discrete Morse function, we still need to make sure that intervals in the same level set are *separated*, by which we mean that no simplex of one interval is face of a simplex in the other interval. By assumption of general position, there is only one level set that contains more than one interval, namely $\mathcal{R}^{-1}(0)$. All its intervals are of the form $[y_x, 0y_x]$, in which x is a point in X, the origin $0 \in \mathbb{R}^{d+1}$ corresponds to the three-partition $(\emptyset, \emptyset, X)$, and $0y_x$ is the edge that connects 0 with y_x. While these edges all share 0, no two also share the other endpoint. It follows that these intervals are components of the Hasse diagram of the level set, as required. ◀

The radius function on a Delaunay mosaic. Recall that the order-k Delaunay mosaic of X is the horizontal slice of the rhomboid tiling at depth k . In other words, every cell of $\mathrm{Del}_k(X)$ is the horizontal slice of a rhomboid. More formally, for every $\sigma \in \mathrm{Del}_k(X)$ there is a unique lowest-dimensional rhomboid $\rho \in \mathrm{Rho}(X)$ such that $\sigma = \rho \cap P_k$, in which P_k is the horizontal d-plane defined by $z = -k$. For vertices we have $\dim \sigma = \dim \rho = 0$, and for all higher-dimensional cells we have $\dim \sigma = \dim \rho - 1 \geq 1$. The *radius function* on the order-k Delaunay mosaic, $\mathcal{R}_k \colon \mathrm{Del}_k(X) \to \mathbb{R}$, is simply the restriction of $\mathcal{R}$ to the horizontal slice: $\mathcal{R}_k(\sigma) = \mathcal{R}(\rho)$. Importantly, this definition is consistent with the subcomplexes $\mathrm{Del}_k(X, r) \subseteq \mathrm{Del}_k(X)$ used to represent the k-fold cover of X and r, but this needs a proof.

▶ **Lemma 4** (Delaunay Radius Function)**.** *Let $X \subseteq \mathbb{R}^d$ be locally finite and in general position. For every integer $k \geq 1$ and every real r, we have* $\mathrm{Del}_k(X, r) = \mathcal{R}_k^{-1}[-\infty, r]$.

Proof. Recall that $\pi_t \colon \mathbb{R}^d \to \mathbb{R}$ is defined by $\pi_t(p) = \frac{1}{2}(\|p\|^2 - t)$. The graph of π_t is a paraboloid that intersects $\mathbb{R}^d$ in the sphere with squared radius t. More generally, the paraboloid intersects every d-plane tangent to the graph of π_0 in an ellipsoid whose vertical projection to $\mathbb{R}^d$ is a sphere with squared radius t. Dropping the paraboloid vertically thus translates into growing balls simultaneously and uniformly centered at the points in X. By definition, $\mathcal{R}(\rho)$ is the value t_0 of t for which the paraboloid touches the dual cell, $\rho^* \in \mathrm{Arr}(X)$, for the first time. More formally, the set of points $q \in \rho^*$ that lie on or above the graph of π_t is empty for all $t < t_0$ and non-empty for all $t \geq t_0$.

Let σ^* be the vertical projection of ρ^* to $\mathbb{R}^d$, and assume it is a polyhedron in some Voronoi tessellation of X. It belongs to $\mathrm{Vor}_k(X)$ iff its dual cell, σ, belongs to $\mathrm{Del}_k(X)$ or, equivalently, if $\mathcal{R}_k(\sigma)$ is defined. Assuming the latter, $\sigma^* \cap \mathrm{Cover}_k(X, r)$ is empty for all $r < r_0$ and non-empty for all $r \geq r_0$, in which $r_0^2 = t_0 = \mathcal{R}(\rho) = \mathcal{R}_k(\sigma)$. By definition, σ belongs to $\mathrm{Del}_k(X, r)$ iff this intersection is non-empty, which implies $\mathrm{Del}_k(X, r) = \mathcal{R}_k^{-1}[-\infty, r]$ for all $r \in \mathbb{R}$, as required. ◀

These results facilitate the computation of the persistence of the k-fold covers for varying radii. Lemma 2 asserts that we can use $\mathrm{Del}_k(X, r)$ as a proxy for $\mathrm{Cover}_k(X, r)$. Lemma 4

provides the recipe for computing the radii of the cells of $\mathrm{Del}_k(X)$, and thus the sublevel set filtration of $\mathrm{Del}_k(X)$, whose persistence module is isomorphic to the persistence of $\mathrm{Cover}_k(X,r)$ for varying radius r. Finally, the persistence diagram is obtained from the filtration via the boundary matrix reduction algorithm [8, Chapter VII].

Assuming $X \subseteq \mathbb{R}^d$ is locally finite and in general position, the radius function of the order-1 Delaunay mosaic is known to be a generalized discrete Morse function [3]. This property does not generalize to higher order. Nevertheless, we can still classify the steps of $\mathcal{R}_k$ into critical and non-critical types such that each critical step changes the homotopy type of the sublevel set in a predictable manner, and every non-critical step maintains the homotopy type of the sublevel set. The proof of this claim together with an enumeration of the types of steps can be found in [9].

4 Persistence in depth

In this section, we develop an algorithm that computes the persistence of the nested sequence of multi-covers (5). We follow the usual strategy of substituting a complex for each cover, but there are complications. Specifically, we represent $\mathrm{Cover}_k(X,r)$ by $\mathrm{Del}_k(X,r)$ and we introduce additional complexes between contiguous Delaunay mosaics to realize the inclusion between the covers.

Half-integer slices. There are generally no convenient maps connecting $\mathrm{Del}_k(X)$ with $\mathrm{Del}_{k-1}(X)$. To finesse this difficulty, we use the horizontal *half-integer slice* of the rhomboid tiling at depth $\ell = k - \frac{1}{2}$:

$$\mathrm{Del}_\ell(X) = \mathrm{Rho}(X) \cap P_\ell, \tag{7}$$

for $k \geq 1$. Similar to the Delaunay mosaic, the half-integer slice is a regular complex in $\mathbb{R}^d$. Not surprisingly, there is a well-known dual, namely the *degree-k Voronoi tessellation* [10], which we denote $\mathrm{Vor}_\ell(X)$. It refines the order-k Voronoi tessellation by decomposing its domains into maximal regions in which every point has the same k-th nearest point in X. Similarly, the degree-k tessellation refines the order-$(k-1)$ tessellation, and indeed $\mathrm{Vor}_\ell(X)$ is the superposition of $\mathrm{Vor}_k(X)$ and $\mathrm{Vor}_{k-1}(X)$. It can be constructed by projecting the k-th level in $\mathrm{Arr}(X)$ to $\mathbb{R}^d$. Without going into further details, we observe that this level contains every cell of the arrangement whose corresponding ordered three-partition, $X = X_{in} \cup X_{on} \cup X_{out}$, satisfies $|X_{in}| \leq k-1$ and $|X_{in}| + |X_{on}| \geq k$. We refer to the decomposition of $\mathrm{Cover}_k(X,r)$ by $\mathrm{Vor}_\ell(X)$ as $\mathrm{Vor}_\ell(X,r)$.

Returning to the mosaics, there are natural piecewise linear maps from $\mathrm{Del}_\ell(X)$ to $\mathrm{Del}_k(X)$ and to $\mathrm{Del}_{k-1}(X)$. Specifically, we get $\mathrm{Del}_\ell(X) \to \mathrm{Del}_k(X)$ by mapping the vertices dual to the regions decomposing $\mathrm{dom}(Q) \in \mathrm{Vor}_k(X)$ to the vertex dual to $\mathrm{dom}(Q)$. Symmetrically, we get $\mathrm{Del}_\ell(X) \to \mathrm{Del}_{k-1}(X)$. However, because such maps lead to complications in the persistence algorithm, we use the horizontal slabs of the rhomboid tiling to connect the mosaics via inclusions. To formally define them, write P_ℓ^k for the points in $\mathbb{R}^{d+1}$ that lie on or between P_ℓ and P_k. We define *slab mosaics* as intersections of such slabs with the rhomboid tiling. Analogous to $\mathrm{Del}_k(X,r)$, we also define radius-dependent subcomplexes of these slab mosaics, as well as of half-integer mosaics:

$$\mathrm{Del}_\ell^k(X,r) = \{\rho \cap P_\ell^k \mid \mathcal{R}(\rho) \leq r\}, \tag{8}$$

$$\mathrm{Del}_\ell(X,r) = \{\rho \cap P_\ell \mid \mathcal{R}(\rho) \leq r\}, \tag{9}$$

$$\mathrm{Del}_{k-1}^\ell(X,r) = \{\rho \cap P_{k-1}^\ell \mid \mathcal{R}(\rho) \leq r\}. \tag{10}$$

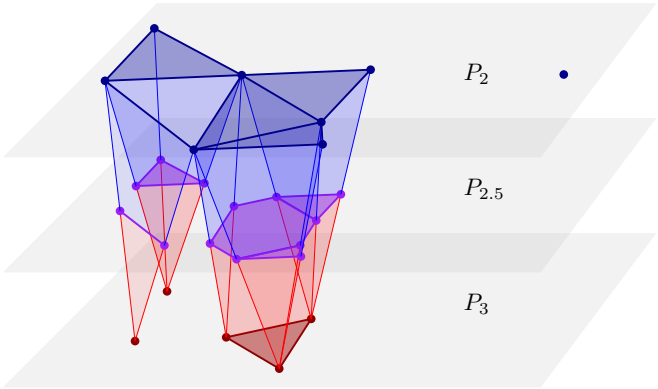


Figure 5 A sublevel set of the 3-dimensional rhomboid tiling of the points in Figure 4. From *top* to *bottom*: D_2 in dark blue, $D_{2.5}$ in purple, and D_3 in dark red, with slabs connecting adjacent slices.

To simplify the notation, we fix r and write $D_k = \mathrm{Del}_k(X, r)$, $C_k = \mathrm{Cover}_k(X, r)$, etc. The half-integer Delaunay mosaic includes in both slab mosaics, D_k includes in the first, and D_{k-1} includes in the second; see Figure 5. We note an important difference between the two slabs: $\mathrm{Vor}_\ell(X, r)$ and $\mathrm{Vor}_k(X, r)$ are different convex subdivisions of the same space, C_k, which implies that D_ℓ and D_k have the same homotopy type. Indeed, this is also the homotopy type of D^k_ℓ, and there are natural deformation retractions to D_ℓ and D_k. In contrast, D_{k-1} and D_ℓ have generally different homotopy types, and there is a deformation retraction from D^ℓ_{k-1} to D_{k-1} but not necessarily to D_ℓ; see again Figure 5. To remedy this deficiency, we introduce mosaics that contain D_ℓ and D^ℓ_{k-1} as subcomplexes. To construct them, we recall that $\mathrm{Vor}_\ell(X)$ is a refinement of $\mathrm{Vor}_{k-1}(X)$, which implies that the polyhedra of $\mathrm{Vor}_\ell(X)$ intersect C_{k-1} in convex sets. We let E_ℓ be the dual of this convex decomposition of the $(k-1)$-fold cover. Since $C_k \subseteq C_{k-1}$, we indeed have $D_\ell \subseteq E_\ell$; see the right panel in Figure 4. Furthermore, we let E^ℓ_{k-1} be the maximal subcomplex of $\mathrm{Rho}(X) \cap P^\ell_{k-1}$ whose boundary complexes at depths $k-1$ and ℓ are D_{k-1} and E_ℓ. Clearly, D^ℓ_{k-1} is a subcomplex of E^ℓ_{k-1}, and because D_{k-1} and E_ℓ are deformation retracts of E^ℓ_{k-1}, these three mosaics have the same homotopy type. We will use these relations shortly in the computation of the persistence diagram of the filtration of multi-covers (5).

Connecting the spaces. To prepare the construction of the persistence and zigzag modules, we connect the multi-covers and the corresponding Delaunay mosaics with maps. Fixing $r \geq 0$ and setting $\ell = k - \frac{1}{2}$, as before, we consider the following diagram in which identities and homotopy equivalences are marked as such:

$$\begin{array}{ccccccccccccc}
\longrightarrow & C_k & \xrightarrow{\mathrm{id}} & C_k & \xrightarrow{\mathrm{id}} & C_k & \xrightarrow{\mathrm{id}} & C_k & \xrightarrow{\mathrm{id}} & C_k & \longrightarrow & C_{k-1} & \xrightarrow{\mathrm{id}} \\
 & \uparrow{\scriptstyle \mathrm{he}} & & \uparrow{\scriptstyle \mathrm{he}} & & \uparrow{\scriptstyle \mathrm{he}} & & \uparrow{\scriptstyle \mathrm{he}} & & \uparrow{\scriptstyle \mathrm{he}} & & \uparrow{\scriptstyle \mathrm{he}} & \\
\longrightarrow & E_{\ell+1} & \xrightarrow{\mathrm{he}} & E^{\ell+1}_k & \xleftarrow{\mathrm{he}} & D_k & \xrightarrow{\mathrm{he}} & D^k_\ell & \xleftarrow{\mathrm{he}} & D_\ell & \longrightarrow & E_\ell & \xrightarrow{\mathrm{he}}
\end{array}$$

The top row stretches out the filtration by writing each multi-cover five times and connecting the copies with the identity. The remaining maps in this row are inclusions. The bottom row contains the slice mosaics at integer and half-integer depths, and connects them with inclusions, using slab mosaics as intermediaries. As argued above, the first five mosaics all have the same homotopy type, and the inclusion maps between them are homotopy equivalences.

To get the vertical map from D_k to C_k, we first construct the barycentric subdivision, $\operatorname{Sd} D_k$, which is a simplicial complex. Each vertex $u \in \operatorname{Sd} D_k$ represents a j-cell in D_k, which is dual to a $(d-j)$-dimensional Voronoi polyhedron, and we map u to the center of mass of the intersection of this polyhedron with the k-fold cover. By construction, this intersection is non-empty and convex, so it contains the center of mass in its interior. After mapping all vertices, we map the other simplices of $\operatorname{Sd} D_k$ by piecewise linear interpolation; see the middle panel in Figure 4. The resulting map is injective, and since D_k and C_k have the same homotopy type, the map is a homotopy equivalence. Recall that D_ℓ is dual to $\operatorname{Vor}_\ell(X, r)$, which is another convex decomposition of the k-fold cover. We therefore get the vertical map from D_ℓ to C_k the same way, first constructing $\operatorname{Sd} D_\ell$ and second mapping the vertices to centers of mass. This is again a homotopy equivalence. Similarly, E_ℓ is dual to the convex decomposition of C_{k-1} with $\operatorname{Vor}_\ell(X)$. As before, we get the vertical map by sending the vertices of E_ℓ to centers of mass, but we distinguish between two cases. If a polyhedron of $\operatorname{Vor}_\ell(X)$ has a non-empty intersection with C_k, we send the corresponding vertex of $\operatorname{Sd} E_\ell$ to the center of mass of this intersection. If, however, the intersection with C_k is empty but the intersection with C_{k-1} is non-empty, then we send the vertex to the center of mass of the latter. This ensures that the geometric embedding of $\operatorname{Sd} D_\ell$ is contained in the geometric embedding of $\operatorname{Sd} E_\ell$.

To finally map the slab mosaics, we first deformation retract them to slice mosaics and then map them reusing the barycentric subdivisions. Here we make arbitrary choices, mapping $E_k^{\ell+1}$ to $E_{\ell+1}$ to C_k and mapping D_ℓ^k to D_k to C_k. Note that all vertical maps are homotopy equivalences, as marked in the above diagram.

Modules. Applying the homology functor for a fixed coefficient field, we map all multi-covers and mosaics to vector spaces and all maps to homomorphisms (linear maps) between them. The top row of vector spaces with homomorphisms from left to right is referred to as a *persistence module*, and we denote it $\mathsf{MC}(r)$. The bottom row of vector spaces are connected by homomorphisms going from left to right or from right to left. This kind of structure is referred to as a *zigzag module*, and we denote it $\mathsf{ZZ}(r)$. The advantage of the zigzag over the persistence module is that its maps are induced by inclusions between complexes, which lend themselves to computations. Our goal, however, is to compute the persistence diagram of $\mathsf{MC}(r)$, and we do this by using $\mathsf{ZZ}(r)$ as a proxy. The following result is therefore essential.

▶ **Lemma 5** (Isomorphism of Modules). *Let $X \subseteq \mathbb{R}^d$ be locally finite and in general position. Then the persistence diagrams of $\mathsf{MC}(r)$ and of $\mathsf{ZZ}(r)$ are the same for every $r \geq 0$.*

Proof. Write $\mathsf{C}_k, \mathsf{D}_k, \mathsf{E}_k$ for the vector spaces obtained by applying the homology functor to C_k, D_k, E_k, etc. The goal is to show that the diagram of multi-covers and mosaics maps to a diagram of vector spaces in which all squares commute and most maps are isomorphisms:

$$\begin{array}{ccccccccccccc}
\longrightarrow & \mathsf{C}_k & \xrightarrow{\text{iso}} & \mathsf{C}_k & \xrightarrow{\text{iso}} & \mathsf{C}_k & \xrightarrow{\text{iso}} & \mathsf{C}_k & \xrightarrow{\text{iso}} & \mathsf{C}_k & \longrightarrow & \mathsf{C}_{k-1} & \xrightarrow{\text{iso}} \\
 & \uparrow{\scriptstyle\text{iso}} & & \uparrow{\scriptstyle\text{iso}} & & \uparrow{\scriptstyle\text{iso}} & & \uparrow{\scriptstyle\text{iso}} & & \uparrow{\scriptstyle\text{iso}} & & \uparrow{\scriptstyle\text{iso}} & \\
\longrightarrow & \mathsf{E}_{\ell+1} & \xrightarrow{\text{iso}} & \mathsf{E}_k^{\ell+1} & \xleftarrow{\text{iso}} & \mathsf{D}_k & \xrightarrow{\text{iso}} & \mathsf{D}_\ell^k & \xleftarrow{\text{iso}} & \mathsf{D}_\ell & \longrightarrow & \mathsf{E}_\ell & \xrightarrow{\text{iso}}
\end{array}$$

To prove commutativity, we consider the five squares shown in the above diagram. The first square commutes already before applying the homology functor, and so does the third square. Similarly, the fifth square commutes because the image of $\operatorname{Sd} D_\ell$ in C_k includes in the image of $\operatorname{Sd} E_\ell$ in C_{k-1}.

The second and fourth squares do not commute before applying the homology functor, but we argue they do after applying the functor. The two cases are similar, so we focus on the fourth square. Recall that $\mathrm{Vor}_k(X,r)$ and $\mathrm{Vor}_\ell(X,r)$ are two convex decompositions of the same space, which is C_k, and that $\mathrm{Vor}_\ell(X,r)$ is a refinement of $\mathrm{Vor}_k(X,r)$. D_k and D_ℓ are dual to these decompositions, with one or more vertices of D_ℓ corresponding to every one vertex of D_k. When we map D_ℓ to D_ℓ^k to C_k, the full subcomplex with these vertices is first contracted to the single vertex by the deformation retraction from D_ℓ^k to D_k, and second it is mapped to the center of mass of the corresponding domain in $\mathrm{Vor}_k(X,r)$. In contrast, when we map D_ℓ to C_k directly, all these vertices map to different points in C_k, but all these points lie in the interior of the same domain in $\mathrm{Vor}_k(X,r)$. Indeed, the full subcomplex with these vertices is dual to a convex decomposition of this domain and therefore contractible. It follows that the fourth square of homomorphisms commutes. Similarly, the second square commutes, and therefore all squares commute.

Isomorphisms are reversible, so we can draw them from left to right in the bottom row of the diagram. The result are two parallel persistence modules whose vector spaces are connected by isomorphisms. The Persistence Equivalence Theorem of persistent homology [8, page 159] implies that the two modules have the same persistence diagram. ◀

Algorithm and running time. We compute the persistence diagram of the filtration of multi-covers (5) using the zigzag algorithm generically described in [4] and explained in detail for inclusion maps in [5]. Its worst-case running time is cubic in the input size, which is the total number of cells in the mosaics. To count the cells, we assume a finite number of points in $\mathbb{R}^d$, $n = |X|$. All cells are horizontal slices or horizontal slabs of rhomboids in $\mathbb{R}^{d+1}$. We therefore begin by counting the rhomboids or, equivalently, the cells in the dual hyperplane arrangement. These numbers are maximized when the n hyperplanes are in general position, and then they depend only on n and d. Observe first that for every $0 \leq i \leq d+1$, there are $\binom{n}{d+1-i}$ i-planes, each the common intersection of $d+1-i$ hyperplanes. There is one chamber for each plane, which implies that the number of chambers in the arrangement is

$$\Gamma_{d+1}^{d+1}(n) = \binom{n}{d+1} + \binom{n}{d} + \ldots + \binom{n}{0} \leq \frac{(n+1)^{d+1}}{(d+1)!}. \tag{11}$$

Indeed, the paraboloid used in the proof of Lemma 3 sweeps out the arrangement and encounters a new chamber whenever it first intersects one of the i-planes, for $0 \leq i \leq d+1$. The inequality on the right-hand side in (11) is easy to prove, by induction or otherwise. To count the i-cells in the arrangement, we observe that each i-plane carries an arrangement of $n-(d+1-i)$ $(i-1)$-planes. We get the number of (i-dimensional) chambers in this arrangement from (11), and multiplying with the number of i-planes, we get the number of i-cells:

$$\Gamma_i^{d+1}(n) = \binom{n}{d+1-i}\Gamma_i^i(n-d-1+i) \leq \frac{n^{d+1-i}}{(d+1-i)!}\frac{(n+1)^i}{i!} \leq \frac{(n+1)^{d+1}}{(d+1-i)!\;i!}. \tag{12}$$

Writing $j = d-i$, we get a $(j+1)$-rhomboids in $\mathrm{Rho}(X)$ for every i-cell in the arrangement. In other words, (12) counts the $(j+1)$-rhomboids in the rhomboid tiling. In particular, we have $\Gamma_{d+1}^{d+1}(n)$ vertices in the tiling. For $0 \leq j \leq d$, the interior of every $(j+1)$-rhomboid has a non-empty intersection with $2j+1$ hyperplanes P_ℓ, in which 2ℓ is an integer. The $(j+1)$-rhomboid thus contributes $2j+1$ j-cells to the Delaunay mosaics and $2j+2$ $(j+1)$-prisms to the slab mosaics. Taking the sum over all dimensions, we get the total number of cells in

the mosaics used in the construction of the zigzag module:

$$\#\text{cells} = \Gamma_{d+1}^{d+1}(n) + \sum_{j=0}^{d}(4j+3)\Gamma_{d-j}^{d+1}(n) \leq \frac{(n+1)^{d+1}}{(d+1)!} + \sum_{i=0}^{d} 4(d{+}1{-}i)\frac{(n+1)^{d+1}}{(d{+}1{-}i)!\; i!} \quad (13)$$

$$\leq \frac{(n+1)^{d+1}}{(d+1)!} + 4(n+1)^{d+1}\sum_{i=0}^{d}\frac{1}{(d-i)!\; i!} \leq 9(n+1)^{d+1}. \quad (14)$$

Taking the third power, we get an upper bound for the worst-case running time of the algorithm and thus the main result of this section.

▶ **Theorem 6** (Multi-cover Persistence). *Let X be a set of n points in general position in $\mathbb{R}^d$. For every radius $r \geq 0$, the persistence diagram of the filtration of multi-covers with radius r can be computed in worst-case time $O(n^{3d+3})$.*

5 Discussion

The main contribution of this paper is the introduction of the $(d+1)$-dimensional rhomboid tiling of a locally finite set of points in $\mathbb{R}^d$. It is the underlying framework that facilitates the study of multi-covers with Euclidean balls and the computation of the persistence as we increase the radius or we decrease the depth of the coverage. The latter requires novel adaptations of the standard approach to persistence, which for n points in $\mathbb{R}^d$ lead to an algorithm with worst-case running time $O(n^{3d+3})$. This compares favorably to naive solutions and the approach using Čech complexes [21], but it is not practical unless n and d are small. While the time-complexity is too high for the density analysis of large data sets, we see applications in the study of regular or semi-regular configurations that arise in the design and investigation of materials. With some modifications, our results extend to balls with different radii (points with weights); see [9], but the implied loss of intuitive appeal prevents us from discussing this generalization. In particular, Theorem 1 extends, and Theorem 6 holds without change in this more general setting. There are a number of challenging questions raised by the work reported in this paper.

- Instead of computing the persistence in scale and depth separately, it might be interesting to combine both to a concrete setting for 2-parameter persistence [18].
- A set of n points in $\mathbb{R}^d$ has some constant times n^{d+1} ordered three-partitions defined by spheres. We cannot improve the worst-case time of our persistence in depth algorithm unless we avoid the enumeration of these partitions. Can this be done?

As proved in [1], for every locally finite $X \subseteq \mathbb{R}^d$, there is a locally finite $Y \subseteq \mathbb{R}^d$ with real weights such that the (order-1) weighted Voronoi tessellation of Y is the order-k Voronoi tessellation of X. However, growing balls uniformly with centers in X and growing them according to the weights with centers in Y gives different filtrations of the dual Delaunay mosaic. It would be interesting to quantify this difference by bounding the distance between the two persistence diagrams.

References

1 Franz Aurenhammer. A new duality result concerning Voronoi diagrams. *Discrete & Computational Geometry*, 5(3):243–254, 1990.

2 Franz Aurenhammer and Otfried Schwarzkopf. A simple on-line randomized incremental algorithm for computing higher order Voronoi diagrams. *International Journal of Computational Geometry & Applications*, 2(04):363–381, 1992.

3 Ulrich Bauer and Herbert Edelsbrunner. The Morse theory of Čech and Delaunay complexes. *Trans. Amer. Math. Soc.*, 369(369):3741–3762, 2017.

4 Gunnar Carlsson and Vin de Silva. Zigzag persistence. *Found. Comput. Math.*, 10(4):367–405, 2010.

5 Gunnar Carlsson, Vin De Silva, and Dmitriy Morozov. Zigzag persistent homology and real-valued functions. In *Proceedings of the twenty-fifth annual symposium on Computational geometry*, pages 247–256. ACM, 2009.

6 Frédéric Chazal, David Cohen-Steiner, and Quentin Mérigot. Geometric inference for probability measures. *Found. Comput. Math.*, 11(6):733–751, 2011.

7 Herbert Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer-Verlag, Heidelberg, Germany, 1987.

8 Herbert Edelsbrunner and John L. Harer. *Computational Topology. An Introduction*. American Mathematical Society, Providence, RI, 2010.

9 Herbert Edelsbrunner, Anton Nikitenko, and Georg Osang. A step in the weighted Delaunay mosaic of order k., 2017. Manuscript, IST Austria, Klosterneuburg, Austria.

10 Herbert Edelsbrunner and Raimund Seidel. Voronoi diagrams and arrangements. *Discrete Comput. Geom.*, 1(1):25–44, 1986.

11 G. Fejes Tóth. Multiple packing and covering of the plane with circles. *Acta Math. Acad. Sci. Hungar.*, 27(1-2):135–140, 1976.

12 Robin Forman. Morse theory for cell complexes. *Adv. Math.*, 134(1):90–145, 1998.

13 Ragnar Freij. Equivariant discrete Morse theory. *Discrete Math.*, 309(12):3821–3829, 2009.

14 Leonidas Guibas, Dmitriy Morozov, and Quentin Mérigot. Witnessed k-distance. *Discrete Comput. Geom.*, 49(1):22–45, 2013.

15 Dmitry Krasnoshchekov and Valentin Polishchuk. Order-k α-hulls and α-shapes. *Inform. Process. Lett.*, 114(1-2):76–83, 2014.

16 Der-Tsai Lee et al. On k-nearest neighbor Voronoi diagrams in the plane. *IEEE Trans. Comput.*, 31(6):478–487, 1982.

17 Jean Leray. Sur la forme des espaces topologiques et sur les points fixes des représentations. *J. Math. Pures Appl. (9)*, 24:95–167, 1945.

18 Michael Lesnick and Matthew Wright. Interactive visualization of 2-D persistence modules. *arXiv preprint arXiv:1512.00180*, 2015.

19 Ketan Mulmuley. Output sensitive construction of levels and Voronoi diagrams in $\mathbb{R}^d$ of order 1 to k. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 322–330. ACM, 1990.

20 Michael Ian Shamos and Dan Hoey. Closest-point problems. In *16th Annual Symposium on Foundations of Computer Science (Berkeley, Calif., 1975)*, pages 151–162. IEEE Computer Society, Long Beach, Calif., 1975.

21 Donald R. Sheehy. A multi-cover nerve for geometric inference. In *Proc. Canadian Conf. Comput. Geom.*, 2012.

Smallest Enclosing Spheres and Chernoff Points in Bregman Geometry

Herbert Edelsbrunner
IST Austria (Institute of Science and Technology Austria)
Am Campus 1, 3400 Klosterneuburg, Austria
edels@ist.ac.at

Žiga Virk
Faculty of Mathematics and Physics, University of Ljubljana
Jadranska ulica 19, 1000 Ljubljana, Slovenia
ziga.virk@fmf.uni-lj.si

Hubert Wagner
IST Austria (Institute of Science and Technology Austria)
Am Campus 1, 3400 Klosterneuburg, Austria
hwagner@ist.ac.at

Abstract

Smallest enclosing spheres of finite point sets are central to methods in topological data analysis. Focusing on Bregman divergences to measure dissimilarity, we prove bounds on the location of the center of a smallest enclosing sphere. These bounds depend on the range of radii for which Bregman balls are convex.

2012 ACM Subject Classification Theory of computation → Computational geometry, Mathematics of computing → Combinatoric problems, Mathematics of computing → Algebraic topology

Keywords and phrases Bregman divergence, smallest enclosing spheres, Chernoff points, convexity, barycenter polytopes

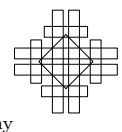
Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.35

Funding This research is partially supported by the Office of Naval Research, through grant no. N62909-18-1-2038, and the DFG Collaborative Research Center TRR 109, 'Discretization in Geometry and Dynamics', through grant no. I02979-N35 of the Austrian Science Fund (FWF).

Acknowledgements The authors thank Arseniy Akopyan for fruitful discussions and an anonymous reviewer for his or her comments on the proof of Theorem 8 and the construction in Section 6.

1 Introduction

Interpreting non-geometric data geometrically is a standard step in data analysis. Examples are abundant, including images [8], medical records [17], text documents [9], and speech samples [4]. The motivating reason for this reinterpretation of data is the availability of standard mathematical tools for multi-dimensional point sets, such as cluster analysis, nearest neighbor search, dimension reduction, data visualization etc. These tools rely on a notion of dissimilarity between data points, and the Euclidean distance is often not ideal. Keeping in mind that a point often represents a histogram describing the corresponding non-geometric object, this is not surprising. A popular alternative to the Euclidean distance is the *Kullback–Leibler divergence*, also known as the *relative entropy* [12], which is built on information theoretic foundations and meaningfully compares probability distributions.




There is experimental evidence for its efficacy, and this in spite of violating two of the three axioms we require from a metric; see [9] for a comparison of measures used to cluster text documents. The relative entropy belongs to the family of *Bregman divergences* [3]. Another member of this family is the Itakura-Saito divergence, which is classically used to compare power spectra of speech patterns [10]. The extension of topological data analysis methods from the Euclidean metric to Bregman divergences needs smallest enclosing spheres to turn data into Bregman–Čech complexes, and smallest circumspheres to turn data into Bregman–Delaunay complexes. We are therefore motivated to study these spheres in detail.

Notation and terminology. We introduce the most important concepts studied in this paper before reviewing prior work and presenting our results. A function $F\colon \Omega \to \mathbb{R}$ on an open convex subset $\Omega \subseteq \mathbb{R}^d$ is of *Legendre type* if

- F is strictly convex,
- F is differentiable,
- the length of the gradient goes to infinity when we approach the boundary of Ω.

The combination of convexity and differentiability implies continuous differentiability; see [5, Theorem 2.86]. The somewhat technical third condition guarantees that the conjugate of F is also of Legendre type; see [18, page 259]. There will be no appearance of the conjugate in this paper, but we will make use of a consequence of the conjugate being of Legendre type proved in [7]. The *Bregman divergence* from x to y associated with F is the difference between F and the best linear approximation of F at y, both evaluated at x:

$$D_F(x\|y) = F(x) - [F(y) + \langle \nabla F(y), x - y\rangle]. \tag{1}$$

The divergence is not necessarily symmetric. We therefore define two balls with given center and radius, one by measuring the divergence *from* the center and the other *to* the center. Specifically, the *primal* and *dual Bregman balls* with center $x \in \Omega$ and radius $r \geq 0$ are

$$B_F(x;r) = \{y \in \Omega \mid D_F(x\|y) \leq r\}, \tag{2}$$
$$B^*_F(x;r) = \{y \in \Omega \mid D_F(y\|x) \leq r\}. \tag{3}$$

While the dual ball is necessarily convex, this is not true for the primal ball. Since we use F throughout this paper, we will feel free to drop it from the notation. An *enclosing sphere* of a set $X \subseteq \Omega$ is the boundary of a dual Bregman ball that contains all points of X. A *circumsphere* of X is an enclosing sphere that passes through all points of X.

Prior work and results. The family of Bregman divergences is named after Lev Bregman who studied convex programming problems in [3]. Each such divergence is based on a Legendre type function; see Rockafellar [18]. A prominent member of the family is the relative entropy, which is based on the Shannon entropy. Its introduction by Kullback and Leibler [12] predates the work of Bregman. Boissonnat, Nielsen, and Nock pioneered the study of Bregman divergences within the fields of computational and information geometry. In [15, 16] they studied algorithms for fitting Bregman balls enclosing a set of points, and in [1] they introduced Bregman–Voronoi diagrams. To get a useful dual structure, we need the non-empty common intersections of primal Bregman balls with the corresponding Voronoi domains be contractible, a property proved in [7]. This opened the door to constructing filtrations of Bregman–Čech and Bregman–Delaunay complexes and to analyzing the data with persistent homology, which is one of the key tools in topological data analysis.

The bridge to the work in this paper is the observation that a collection of primal Bregman balls of radius r have a non-empty common intersection iff their centers are contained in a

dual Bregman ball of the same radius r. In this paper, we study the location of the center of the smallest enclosing sphere of a finite set of points, and we follow [14] in calling this center the *Chernoff point* of the set. It is easy to show that the Chernoff point belongs to the convex hull of the finite set, which was first proven in [16]. For completeness, we present a proof of this observation based on the widely used Bregman–Pythagoras Theorem; see e.g. [1]. To improve on this insight, we distinguish between Bregman divergences with convex and with nonconvex balls. The original contributions presented in this paper are:

- for convex balls, we show that the Chernoff point of a simplex is contained in the convex hull of the Chernoff points of its facets, which is generally a much smaller space of possible locations;
- for nonconvex balls, we prove a weaker result with heavier machinery.

To provide context for these results, we mention that this paper follows [6, 7] as third in a series. The broader goal is to lay the theoretical foundations needed to expand the range of applications in which topological tools can be meaningfully used. This line of research established that Bregman divergences and the balls they induce are compatible with methods from computational topology, and in particular with persistent homology. The initial steps in this direction left however many important questions unanswered. In this undertaking, the location of the Chernoff point of a set plays a crucial role. A limiting factor was the general paucity of nontrivial bounds on its location.

The new bounds are useful, for example, in pruning redundant computations when Chernoff points of many small and possibly overlapping point sets have to be computed – a scenario that is typical for topological constructions. This contrasts the usual setting in which the computation of the Chernoff point for a single and possibly large point set is considered.

In summary, we believe that a deeper understanding of the often counterintuitive behavior of Bregman divergences is needed to reach the full potential of the topological tools. This paper contributes by demonstrating that ideas form combinatorial topology are useful in the study of Chernoff points in particular and of Bregman divergences in general.

Outline. Section 2 proves basic properties of smallest enclosing spheres and smallest circumspheres. Section 3 introduces barycenter polytopes. Section 4 proves our main result in the easier convex case. Section 5 extends the main result to the more difficult nonconvex case. Section 6 shows that the nesting hierarchy proved in the nonconvex case is best possible. Section 7 concludes this paper.

2 Smallest spheres

Growing primal Bregman balls from given points in Ω, we study the point at which these balls meet first. Equivalently, we study the Chernoff point, which is the center of the smallest enclosing sphere of the given points. In particular, we prove that the Chernoff point of a simplex lies in the simplex, and that the center of the smallest circumsphere lies in the affine hull of the simplex.

Bregman–Pythagoras. We use the following notation throughout this paper: letting $A \subseteq \Omega$ be a closed convex subset and $y \in \Omega$ a point, we write y_A for the point in A that minimizes the Bregman divergence to y: $y_A = \arg\min_{a \in A} D(a\|y)$. We will make use of an extension of Pythagoras' Theorem to Bregman divergences; see e.g. [1]. We give a proof for completeness.

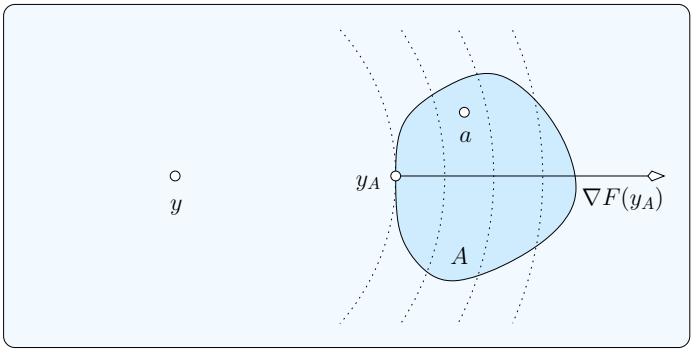


Figure 1 The gradient at a point of a level set of F forms a right angle with the level set.

▶ **Proposition 1** (Bregman–Pythagoras). *All points a of a closed convex set $A \subseteq \Omega$ satisfy $D(a\|y) \geq D(a\|y_A) + D(y_A\|y)$, with equality if A is an affine subspace.*

Proof. We first assume that $F(y) = 0$ and F has its minimum at $y \in \Omega$. It follows that $\nabla F(y) = 0$ and $D(x\|y) = F(x) \geq 0$ for every $x \in \Omega$. The sets of constant distance r to y are therefore the level sets, $F^{-1}(r)$; see Figure 1. The point y_A is where the lowest level set touches A. The gradient of F at y_A is normal to this level set. Hence,

$$F(a) \geq F(a) - \langle \nabla F(y_A), a - y_A \rangle = D(a\|y_A) + F(y_A) \tag{4}$$

because the scalar product is necessarily non-negative. Substituting $F(a) = D(a\|y)$ and $F(y_A) = D(y_A\|y)$, we get $D(a\|y) \geq D(a\|y_A) + D(y_A\|y)$, as claimed. If A is an affine subspace of $\mathbb{R}^d$, then the scalar product in (4) vanishes for all $a \in A$, which implies equality, again as claimed.

If F does not satisfy the simplifying assumption, then we construct $G\colon \Omega \to \mathbb{R}$ defined by $G(x) = F(x) - [F(y) + \langle \nabla F(y), x - y \rangle]$. It is clear that $G(y) = \nabla G(y) = 0$. For any two points $u, v \in \Omega$, we have $D_G(u\|v) = D_F(u\|v)$, so we get the claimed inequality from $D_G(a\|y) \geq D_G(a\|y_A) + D_G(y_A\|y)$, which is implied by the above argument. ◀

Smallest enclosing sphere. Recall that an enclosing sphere of a set $X \subseteq \Omega$ is the boundary of a dual Bregman ball that contains X. We are interested in the smallest such sphere in the case in which X is a set of $k + 1 \leq d + 1$ points. We refer to such a set X as an *(abstract) k-simplex*, and we write $\mathrm{conv}(X)$ for the corresponding geometric k-simplex.

▶ **Lemma 2** (Smallest Enclosing Sphere). *The Chernoff point of any k-simplex $X \subseteq \Omega$ is unique and contained in* $\mathrm{conv}(X)$*, for every $0 \leq k \leq d$.*

Proof. Let $B^*(y; r)$ be a dual Bregman ball with smallest radius that contains all points of X. Writing $x_0, x_1, \ldots, x_k$ for the points in X, this implies $D(x_i\|y) \leq r$ for all i, with equality for at least one index i. To get a contradiction, we set $A = \mathrm{conv}(X)$ and assume $y \notin A$. Using Proposition 1, we get $D(x_i\|y) \geq D(x_i\|y_A) + D(y_A\|y)$ for all $0 \leq i \leq k$. Since $D(x_i\|y) \leq r$ and $D(y_A\|y) > 0$, by assumption of y not being in A, this implies $D(x_i\|y_A) < r$ for all $0 \leq i \leq k$. But this contradicts the minimality of $B^*(y; r)$, and we get $y \in A$ as desired. The uniqueness of y follows from the strict convexity of F. ◀

Smallest circumsphere. Recall that a circumsphere of X is the boundary of a dual Bregman ball that passes through all points of X. There may or may not be any such ball whose center is contained in Ω. To simplify the discussion, we restrict ourselves to a case in which such centers are guaranteed to exist, namely when $\Omega = \mathbb{R}^d$ and X is a set of $k+1 \leq d+1$ points in general position in $\mathbb{R}^d$. Note that for $k+1 < d+1$, the circumsphere is not unique, and often the smallest one is of interest. Using Proposition 1, it is not difficult to prove that the center of the smallest circumsphere of X is contained in the affine hull of X.

▶ **Lemma 3** (Smallest Circumsphere). *The center of the smallest circumsphere of any k-simplex $X \subseteq \mathbb{R}^d$ is unique and contained in* aff X*, for every $0 \leq k \leq d$.*

Proof. Let $B^\circ(y;r)$ be the ball bounded by a smallest circumsphere of X. Writing $x_0, x_1, \ldots, x_k$ for the points in X, this implies $D(x_0\|y) = D(x_1\|y) = \ldots = D(x_k\|y) = r$, and that r is the smallest real number for which there is a point $y \in \mathbb{R}^d$ such that these equalities are satisfied; see Figure 2. To get a contradiction, we set $A = \operatorname{aff} X$ and assume $y \notin A$. Using Proposition 1 for affine subspaces, we get $D(x_i\|y) = D(x_i\|y_A) + D(y_A\|y)$ for all $0 \leq i \leq k$. Because $D(y_A\|y) > 0$, by assumption of y not being in A, this implies $D(x_0\|y_A) = D(x_1\|y_A) = \ldots = D(x_k\|y_A) < r$, which contradicts the minimality of $B^\circ(y;r)$. We get uniqueness because there is only one point in $A = \operatorname{aff} X$ equally far from all the x_i. ◀

3 Barycenter polytopes

Given a simplex, we introduce the family of convex hulls of the face barycenters. The motivation for the study of these polytopes is the sharpening of Lemma 2.

Nested sequence of polytopes. Let $X = \{x_0, x_1, \ldots, x_k\}$ be a k-simplex in $\mathbb{R}^d$. For every subset $J \subseteq \{0, 1, \ldots, k\}$, we write $X_J \subseteq X$ for the corresponding face, $j = |J| - 1$ for the dimension of X_J, and $b_J = \frac{1}{j+1} \sum_{i \in J} x_i$ for the *barycenter* of X_J. For $0 \leq j \leq k$, the *j-th barycenter polytope* of X is

$$\Delta_j^k(X) = \operatorname{conv} \{b_J \mid |J| = j+1\}. \tag{5}$$

Note that $\Delta_0^k = \Delta_0^k(X) = \operatorname{conv}(X)$. In three dimensions, Δ_0^3 is a tetrahedron, Δ_1^3 is an octahedron, Δ_2^3 is again a tetrahedron, and Δ_3^3 is a point. It is not difficult to see that the barycenter polytopes are nested.

▶ **Lemma 4** (Nesting). *The barycenter polytopes of any k-simplex satisfy $\Delta_0^k \supseteq \Delta_1^k \supseteq \ldots \supseteq \Delta_k^k$.*

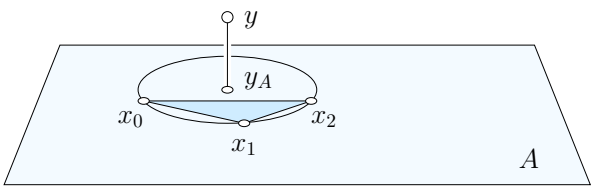


Figure 2 Assuming the center of the smallest circumsphere, y, does not lie in $A = \operatorname{aff} X$ leads to a contradiction.

Proof. Let $J \subseteq \{0, 1, \ldots, k\}$ and assume its cardinality satisfies $j + 1 \geq 2$. We take the average of the barycenters that correspond to J with one index removed:

$$\tfrac{1}{j+1} \sum_{\ell \in J} b_{J \setminus \{\ell\}} = \tfrac{1}{j+1} \sum_{\ell \in J} \left[\tfrac{1}{j} \sum_{i \in J \setminus \{\ell\}} x_i \right] = \tfrac{1}{j(j+1)} \sum_{\ell \in J} \sum_{i \in J \setminus \{\ell\}} x_i. \tag{6}$$

Each point $x_i \in X_J$ appears j times in the double-sum, which implies that the above average is equal to b_J. We thus proved that every vertex of the j-th barycenter polytope is a convex combination of the vertices of the $(j-1)$-th barycenter polytope. Hence, $\Delta^k_j \subseteq \Delta^k_{j-1}$ for $1 \leq j \leq k$, as claimed. ◀

Face structure. It is instructive to take a closer look at Δ^3_1, which is the first barycenter polytope that is not a simplex. Being an octahedron, it has 8 faces of co-dimension one, which we refer to as *facets*. Four of the facets are the 1-st barycenter polytopes of the triangles bounding the tetrahedron, and the other four facets are homothetic copies of the original four triangle. More generally, most barycenter polytopes have twice as many facets as the defining simplex. Write $\#\text{facets}(\Delta^k_j)$ for the number of facets of Δ^k_j.

▶ **Lemma 5** (Number of Facets). *Let $k \geq 1$. The number of facets of the j-th barycenter polytope of a k-simplex is*

$$\#\text{facets}(\Delta^k_j) = \begin{cases} k+1 & \text{if } j = 0, k-1, \\ 2k+2 & \text{if } 1 \leq j \leq k-2. \end{cases} \tag{7}$$

Proof. Index the coordinates of points in $\mathbb{R}^{k+1}$ from 0 to k, and let e_i be the unit vector in the i-th coordinate direction. Identifying the k-simplex with the endpoints of these vectors, we consider the $(k+1)$-dimensional cube spanned by e_0 to e_k and note that this cube has $2k+2$ facets. For $0 \leq j \leq k-1$, we define the *j-th slice* of this cube as the intersection with the k-plane of points $\sum_{i=0}^k \gamma_i e_i$ satisfying $\sum_{i=0}^k \gamma_i = j+1$. It is the convex hull of the $(j+1)$-fold sums of the unit vectors. Scaling the j-th slice by a factor $\frac{1}{j+1}$, we get the j-th barycenter polytope of the k-simplex.

For $j = 0, k-1$, the k-plane intersects half the facets of the cube, and for $1 \leq j \leq k-2$, it intersects all facets of the cube. Each facet of the j-th slice, and after scaling of Δ^k_j, is the intersection of the k-plane with a facet of the cube, which implies the claimed number of facets of the barycenter polytope. ◀

Observe that half the facets of the $(k+1)$-cube share the origin, and the other half share the point $(1, 1, \ldots, 1)$ as a vertex. After scaling, we call the slice of a facet that shares the origin a *far facet* of the corresponding barycenter polytope, noting that it is a barycenter polytope of a facet of the given k-simplex. Similarly after scaling, we call the slice of a facet that shares $(1, 1, \ldots, 1)$ a *near facet* of the barycenter polytope, noting that it is the homothetic copy of a barycenter polytope of a facet of the given k-simplex.

Central projection. For the purpose of the proof of Theorem 8, we subdivide the boundary of Δ^k_j and re-associate the pieces to get the boundary complex of the k-simplex, at least topologically. Write $b(X)$ for the barycenter of the k-simplex, and introduce the central projection,

$$\pi^k_j \colon \partial \text{conv}(X) \to \partial \Delta^k_j, \tag{8}$$

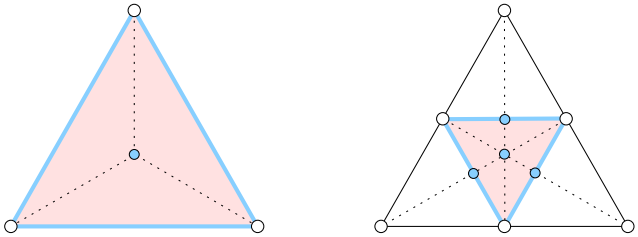


Figure 3 *Left:* the map π_0^2 is the identity on the boundary of the triangle. *Right:* the map π_1^2 projects the vertices of $\mathrm{conv}(X)$ to the midpoints of the edges of Δ_1^2. The structure of $\partial\mathrm{conv}(X)$ is recovered by gluing the half-edges in pairs at the shared endpoints.

which we define by mapping $x \in \partial\mathrm{conv}(X)$ to the unique convex combination of x and $b(X)$ that belongs to the boundary of Δ_j^k. Figure 3 shows the picture of the two maps in the plane.

In the general case, we subdivide $\partial\Delta_j^k$ along the image of the $(k-2)$-skeleton of $\partial\mathrm{conv}(X)$. To convince ourselves that this is well defined, we note that $\partial\Delta_j^k$ is a $(k-1)$-sphere for every $0 \le j \le k-1$. Similarly, $\partial\mathrm{conv}(X)$ is a $(k-1)$-sphere. The center of the projection, $b(X)$, lies in the interior of Δ_j^k and also in the interior of $\mathrm{conv}(X)$, which implies that π_j^k is a homeomorphism. We therefore reach our goal by first glueing the facets of Δ_j^k along their shared faces – which amounts to forgetting the decomposition of the $(k-1)$-sphere these facets imply – and second cutting the $(k-1)$-sphere along the image of the $(k-2)$-skeleton of $\partial\mathrm{conv}(X)$ – which effectively triangulates the $(k-1)$-sphere with $k+1$ $(k-1)$-simplices.

4 Theorem in convex case

This section sharpens Lemma 2 by further limiting the region in which the center of the smallest enclosing sphere can lie. Here we discuss the case in which all primal Bregman balls are convex.

Chernoff polytopes. As before, let X be a k-simplex in Ω. For each subset $J \subseteq \{0, 1, \ldots, k\}$, we recall that X_J is the corresponding face of X, and we let $B^*(d_J; R_J)$ be the smallest dual Bregman ball that contains X_J. Equivalently, $R_J = R(X_J)$ is the minimum radius r such that $\bigcap_{i\in J} B(x_i; r) \neq \emptyset$, and this intersection consists of a single point, namely the Chernoff point d_J of X_J. In analogy with the barycenter polytope of the previous section, we define the *j-th Chernoff polytope* of X as the convex hull of the Chernoff points of faces of dimension $j = |J| - 1$:

$$\Delta_j(X) = \mathrm{conv}\,\{d_J \mid |J| = j+1\} \tag{9}$$

for $0 \le j \le k$; see Figure 4 for an illustration. Note that $\Delta_0(X) = \mathrm{conv}(X)$, but for positive indices j, $\Delta_j = \Delta_j(X)$ is not necessarily the j-th barycenter polytope because the vertices are not necessarily the barycenters of the faces.

Nesting. The Chernoff polytopes drawn in Figure 4 are nested, but that they retain this property of the barycenter polytopes in general needs a proof.

▶ **Theorem 6** (Nesting for Convex Balls). *Let $F\colon \Omega \to \mathbb{R}$ be of Legendre type such that all primal Bregman balls are convex, and let $\Delta_0, \Delta_1, \ldots, \Delta_k$ be the Chernoff polytopes of a k-simplex $X \subseteq \Omega$. Then $\Delta_0 \supseteq \Delta_1 \supseteq \ldots \supseteq \Delta_k$.*

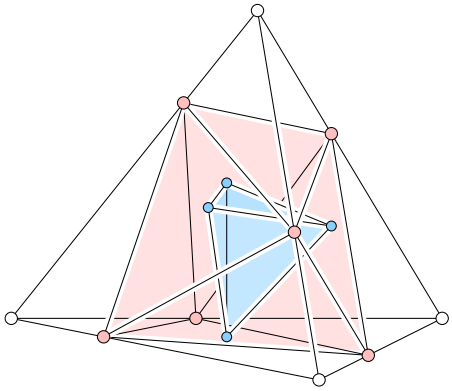


Figure 4 The 1-st Chernoff polytopes of the tetrahedron is the pink octahedron whose vertices lie on the edges of the outer tetrahedron, and the 2-nd Chernoff polytope is the blue tetrahedron whose vertices lie on four of the triangles bounding the octahedron.

Proof. We prove $\Delta_{j-1} \supseteq \Delta_j$ by showing that the Chernoff point of every j-face of $\text{conv}(X)$ is contained in the convex hull of the Chernoff points of the $(j-1)$-faces of this j-face. Indeed, this implies that the vertices of Δ_j lie in Δ_{j-1}, and the claimed inclusion follows. To formalize this idea, let $0 < j \leq k$, set $J = \{0, 1, \ldots, j\}$, and consider $J \setminus \{\ell\}$ for every $0 \leq \ell \leq j$. We prove that d_J belongs to the convex hull of the $d_{J\setminus\{\ell\}}$ in two steps.

For the first step, we write $\text{Sd}(X_J)$ for the *Chernoff subdivision* of $\text{conv}(X_J)$. We obtain it from the barycentric subdivision by moving each barycenter to the location of the corresponding Chernoff point. If all Chernoff points are different, the two subdivisions are isomorphic, but it is possible that two or more barycenters map to the same Chernoff point, in which case some of the simplices in the barycentric subdivision collapse to simplices of smaller dimension. Since $B(x_0; R_J)$ contains the point d_J, it also contains all points $d_{J'}$ with $0 \in J' \subseteq J$. Indeed, if the balls $B(x_i; R_J)$ with $i \in J$ have a point in common, then so do the balls with $i \in J'$. By convexity, $B(x_0; R_J)$ contains all simplices in $\text{Sd}(X_J)$ that share x_0. Similarly, $B(x_\ell; R_J)$ contains all simplices in $\text{Sd}(X_J)$ that share x_ℓ, for each $\ell \in J$. It follows that the $j+1$ balls cover the entire Chernoff subdivision of $\text{conv}(X)$ and thus the entire j-face:

$$\text{conv}(X_J) \subseteq \bigcup_{\ell=0}^{j} B(x_\ell; R_J). \tag{10}$$

For the second step, we write Σ_J for the convex hull of the points $d_{J\setminus\{\ell\}}$, $\ell \in J$. It is the $(j-1)$-st Chernoff polytope of X_J and necessarily contractible. Define $C_\ell = B(x_\ell; R_J) \cap \Sigma_J$, for each $0 \leq \ell \leq j$. By Lemma 2, the points $d_{J\setminus\{\ell\}}$ belong to $\text{conv}(X_J)$, so $\Sigma_J \subseteq \text{conv}(X_J)$, and (10) implies that the sets C_ℓ cover Σ_J. But this does not yet imply that the $(j+1)$-fold intersection of the balls, which is the point d_J, belongs to the C_ℓ and therefore to Σ_J. To prove this, we need the Nerve Theorem [2, 13], which applies to the sets C_ℓ because they are convex. It implies that the nerve of the sets C_ℓ has the same homotopy type as Σ_J and is therefore contractible. The j-fold intersections are all non-empty, as witnessed by the vertices of Σ_J. Hence, the nerve contains the boundary complex of a j-simplex. Contractibility thus implies that the nerve also contains the j-simplex. In other words, $d_J \in C_\ell$ for $0 \leq \ell \leq j$ and therefore $d_J \in \Sigma_J$. ◀

We note that Theorem 6 tightens Lemma 2 in the case of convex Bregman balls: the center of the smallest enclosing sphere of a k-simplex is contained in the convex hull of the centers of the smallest enclosing spheres of all $(k-1)$-faces.

5 Theorem in nonconvex case

We will see shortly that the assumption of convex Bregman balls can be relaxed. The proof of the inclusions in the nonconvex case is the same as in the convex case, except that the individual steps are more complicated. We begin with an auxiliary result.

A fixed point lemma. To generalize Theorem 6 to the nonconvex case, we employ a classic result in topology proved in 1929 by Knaster, Kuratowski, and Mazurkiewicz [11]. It can be used to prove the Brouwer Fixed Point Theorem, which states that every continuous function from the n-dimensional closed ball to itself has a fixed point.

▶ **Proposition 7** (Fixed Point). *Let X be a k-simplex with vertices x_0 to x_k, and let C_0 to C_k be closed sets such that the union of any subcollection of the sets contains the face spanned the corresponding subcollection of vertices. Then $\bigcap_{i=0}^{k} C_i \neq \emptyset$.*

Take for example C_i equal to the closed star of vertex x_i in the barycentric subdivision of $\text{conv}(X)$. The conditions in the proposition are satisfied, and the stars have indeed a non-empty common intersection, namely the barycenter of X. It is important to note that the lemma is topological and therefore also holds for homeomorphically deformed k-simplices.

Interrupted hierarchy. Now suppose that there is a threshold such that all primal Bregman balls with radius at most this threshold are convex, but this is not guaranteed for balls with radius larger than the threshold. An example is the Itakura–Saito divergence [10] defined on the standard simplex whose balls are convex provided the radius does not exceed $\ln 2 - \frac{1}{2} = 0.193\ldots$ [6]. How does this weaken the hierarchy in Theorem 6? To state our claim, we define $R_j = \max_{|J|=j+1} R_J$, noting that $r \geq R_j$ iff all $(j+1)$-fold intersections of the $B(x_i; r)$ are non-empty.

▶ **Theorem 8** (Nesting for Nonconvex Balls). *Let $F\colon \Omega \to \mathbb{R}$ be of Legendre type such that all primal Bregman balls of radius $r \leq R_j$ are convex, and let $\Delta_0, \Delta_1, \ldots, \Delta_k$ be the Chernoff polytopes of a k-simplex $X \subseteq \Omega$. Then $\Delta_0 \supseteq \Delta_1 \supseteq \ldots \supseteq \Delta_j \supseteq \Delta_{j+1}, \Delta_{j+2}, \ldots, \Delta_k$.*

Proof. To prove the inclusions $\Delta_0 \supseteq \Delta_1 \supseteq \ldots \supseteq \Delta_j$, it suffices to consider balls of radius R_j or smaller. These are convex, by assumption, so the inclusions are implied by Theorem 6. To prove $\Delta_j \supseteq \Delta_i$, for $j < i \leq k$, we show that for every i-face, the Chernoff point is contained in the j-th Chernoff polytope. It suffices to consider $i = k$. In the first step of the proof, we generalize (10) to

$$\text{conv}(X_J) \subseteq \bigcup_{\ell \in J} B(x_\ell; R_J) \tag{11}$$

for every $J \subseteq \{0, 1, \ldots, k\}$ also in the nonconvex case. We use induction over the dimension. To simplify the notation, assume $J = \{0, 1, \ldots, j\}$ and let $H = H_J$ be the j-dimensional plane spanned by X_J. By inductive assumption, we have (11) for all $J \setminus \{\ell\}$, $0 \leq \ell \leq j$. By definition of R_J, the $j+1$ balls $B(x_\ell; R_J)$ have a non-empty common intersection, namely the point d_J. By Lemma 2, also the j-dimensional slices of the balls defined by H have the point d_J in common. It follows that the nerve of the sliced balls is a j-simplex, which is contractible. Since F is of Legendre type, so is its restriction to the j-plane, $F|_H$. As proved in [7], this implies that all intersections of the sliced balls are contractible. By the Nerve Theorem [2, 13], the union of the sliced balls is contractible as well. But this union covers the $(j-1)$-dimensional boundary of $\text{conv}(X_J)$, so it must also cover $\text{conv}(X_J)$ to be contractible. Hence (11) follows, and in particular $\text{conv}(X) \subseteq \bigcup_{\ell=0}^{k} B(x_\ell; R_k)$.

For the second step, recall that $\Delta_j = \Delta_j(X)$ is the j-th Chernoff polytope of X. Define $C_\ell = B(x_\ell; R_k) \cap \Delta_j$, for $0 \leq \ell \leq k$. Since $\Delta_j \subseteq \text{conv}(X)$, the balls $B(x_\ell; R_k)$ cover Δ_j. By the Nerve Theorem, the nerve of the sets C_ℓ has the same homotopy type as Δ_j and is therefore contractible. To apply Proposition 7 to Δ_j, we first interpret Δ_j as the homeomorphic image of a k-simplex. Assuming Δ_j is k-dimensional, we decompose its boundary by central projection of the boundary of $\text{conv}(X)$, in which we use any point in the interior of Δ_j as center; see Figures 3 and 4 for illustrations. It is possible that the dimension of Δ_j is less than k, namely when some j-faces of $\text{conv}(X)$ have coincident Chernoff points. We can perturb the coincident Chernoff points slightly and continue the proof with the perturbed Δ_j, which is now k-dimensional. In either case, we denote the topological k-simplex by $\tilde{\Delta}_j$.

Recall that the facets of Δ_j are classified as near and far facets of the $k+1$ points in X. Each facet of $\tilde{\Delta}_j$ consists of a far facet and pieces of k near facets of Δ_j. To describe this in the necessary amount of detail, we denote the facets of $\tilde{\Delta}_j$ by $\Phi_0, \Phi_1, \ldots, \Phi_k$ and the facets of X by $X_\ell = X \setminus \{\ell\}$ for $0 \leq \ell \leq k$. The indexing is chosen so that Φ_ℓ consists of the far facet of x_ℓ – which is contained in $\text{conv}(X_\ell)$ – together with pieces of the near facets of the vertices $x_i \in X_\ell$. The far facet of x_ℓ is covered by the balls $B(x_i; R_k)$, for $i \neq \ell$, as a consequence of (11). Furthermore, the near facet of x_ℓ is covered by $B(x_\ell; R_k)$ simply because $B(x_\ell; R_j) \subseteq B(x_\ell; R_k)$, and the former ball is convex and contains the relevant vertices of Δ_j. It follows that Φ_ℓ is covered by the balls $B(x_i; R_k)$, for $i \neq \ell$. Hence, $\tilde{\Delta}_j$ and the sets C_ℓ satisfy the assumptions of Proposition 7. The proposition thus implies that the common intersection of the C_ℓ is non-empty. This intersection can only be the Chernoff point of X, which we therefore conclude lies inside Δ_j, as required. ◀

In particular, if the balls remain convex until radius R_{k-1}, then Theorem 6 still holds. Without any assumption on convexity, we do not claim anything beyond Δ_0 containing all points d_J, $J \subseteq \{0, 1, \ldots, k\}$, which is Lemma 2.

6 No improvement

We finally show that Theorem 8 is best possible, in the sense that the hierarchy of inclusions cannot be extended beyond Δ_{j+1}. To this end, we construct a function of Legendre type, $F\colon \mathbb{R}^d \to \mathbb{R}$, and a d-simplex, $X \subseteq \mathbb{R}^d$, such that

- for radius $r \leq R_j$, the primal Bregman balls centered at the points of X are convex,
- there is at least one $(j+2)$-face of X whose Chernoff point is not contained in Δ_{j+1}.

It will suffice to consider the case $j + 2 = d$. We begin with the construction in $d = 2$ dimensions, when $j = 0$ and $R_j = 0$, so the first condition is automatically satisfied. With an eye on the generalization to higher dimensions, we will nevertheless make sure that the balls with small but positive radius are convex. We first simplify the task by requiring that F be convex but not necessarily differentiable and not necessarily strictly convex. Appropriate small bump functions are used to eventually turn the convex function into a Legendre type function.

Two-dimensional construction. Let $\Delta_0^2 = \text{conv}(X)$ with $X = \{x_0, x_1, x_2\}$ be an equilateral triangle with edges of length $\sqrt{2}$ and center at the origin in $\mathbb{R}^2$. The first barycenter polytope is $\Delta_1^2 = \text{conv}(Y)$ with $Y = \{y_0, y_1, y_2\}$ and $y_i = \frac{1}{2}(x_{i+1} + x_{i+2})$, where we take indices modulo 3. Calculating the Euclidean inradii of Δ_1^2 and Δ_0^2, we choose a radius strictly between them, $1/\sqrt{24} < \varrho < 1/\sqrt{6}$, and we let $D(\varrho)$ be the (Euclidean) disk with this radius and center at the origin. As illustrated in Figure 5, $D(\varrho)$ is neither contained in Δ_1^2 nor

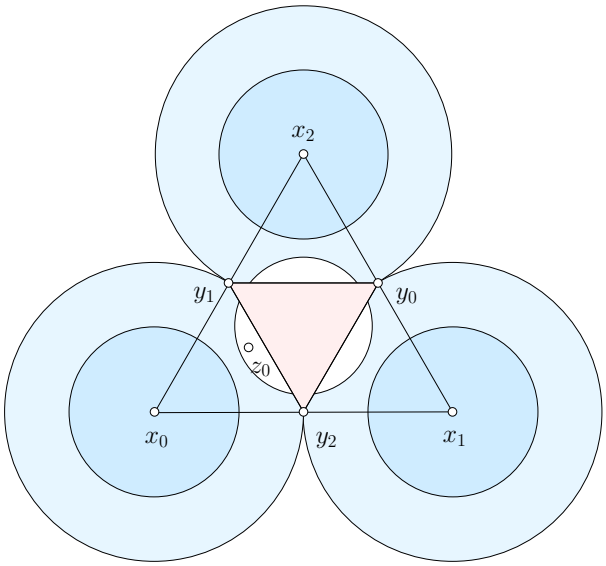


Figure 5 In the nonconvex case, we can arrange that the Chernoff point of X lies outside the triangle spanned by the Chernoff points of the edges of X.

does it contain Δ_1^2. We finally construct $F\colon \mathbb{R}^2 \to \mathbb{R}$ by mapping $a \in \mathbb{R}^2$ to $F(a) = \|a\|^2$ if $a \in \mathbb{R}^2 \setminus D(\varrho)$, and to $F(a) = \varrho^2$ if $a \in D(\varrho)$. The graph of F is a paraboloid with a flattened bottom. Recall that $B_F(x_i; r)$ can be constructed by vertically projecting all points of the graph that are visible from the point $(x_i, \|x_i\|^2 - r) \in \mathbb{R}^2 \times \mathbb{R}$. Writing $D_i(\sqrt{r})$ for the disk with center x_i and squared radius r, the primal Bregman ball satisfies

$$B_F(x_i; r) = \begin{cases} D_i(\sqrt{r}) & \text{if } r \leq (\sqrt{2/3} - \varrho)^2, \\ D_i(\sqrt{r}) \setminus D(\varrho) & \text{if } (\sqrt{2/3} - \varrho)^2 \leq r \leq 2/3 - \varrho^2, \\ D_i(\sqrt{r}) \cup D(\varrho) & \text{if } 2/3 - \varrho^2 < r. \end{cases} \tag{12}$$

The Bregman ball is convex in the first case, and it is nonconvex in the second case. To give the final touch, we observe that the gradient of F is bounded away from zero everywhere outside $D(\varrho)$. We can therefore change F so its graph over $D(\varrho)$ is an upside-down cone with apex $z_0 \in \operatorname{int} D(\varrho) \setminus \Delta_1^2$, and we can do this without violating convexity and without changing F outside this disk. We can turn F into a differentiable and strictly convex function by substituting slightly curved arcs for the generating lines of the cone and by rounding off the sharp corners at the apex and the circle at which the cone meets the paraboloid. With these modifications, we get z_0 as the Chernoff point of X, which by construction lies outside Δ_1^2.

Higher dimensions. The 2-dimensional construction generalizes in a straightforward way to $d \geq 2$ dimensions. The only nontrivial step is to prove that $\varrho > 0$ can be chosen so that the Euclidean ball $D(\varrho)$ neither contains Δ_{d-1}^d nor is contained in it, and that a ball centered at x_i and touching $D(\varrho)$ in a single point contains the near facet of Δ_{d-2}^d. With such a ϱ, we can generalize the 2-dimensional construction so that the Chernoff point of X lies outside Δ_{d-1}^d. We now prove that such a ϱ exists. Let $\Delta_0^d = \operatorname{conv}(X)$ be a regular d-simplex with edges of length $\sqrt{2}$ and center at the origin in $\mathbb{R}^d$. We need formulas for the Euclidean circumradius and height of Δ_0^d, and the Euclidean inradius of Δ_{d-1}^d. It is convenient to derive them for the standard d-simplex, which is the convex hull of the endpoints of the $d+1$

unit coordinate vectors of $\mathbb{R}^{d+1}$. We get the circumradius as the Euclidean distance between the vertices and the center at $(\frac{1}{d+1}, \frac{1}{d+1}, \ldots, \frac{1}{d+1})$:

$$R_d = \sqrt{\left(\frac{d}{d+1}\right)^2 + d\left(\frac{1}{d+1}\right)^2} = \sqrt{\frac{d(d+1)}{(d+1)^2}} = \sqrt{\frac{d}{d+1}}. \tag{13}$$

This radius is $d/(d+1)$ times the height of the standard simplex, which implies that the height is $H_d = (d+1)R_d/d = \sqrt{(d+1)/d}$. To compute the inradius of Δ^d_{d-1}, we observe that the Euclidean distance of the center of Δ^d_0 from a facet is $H_d/(d+1)$. Similarly, the Euclidean distance between parallel facets of Δ^d_{d-1} and Δ^d_0 is H_d/d. It follows that the inradius is

$$I_d = \left[\frac{1}{d} - \frac{1}{d+1}\right] H_d = \frac{1}{d(d+1)} H_d. \tag{14}$$

Consider the Euclidean ball with center x_i and radius $R_d - I_d$. By construction, it touches the $(d-1)$-st barycenter polytope of X at the center of one of its facets, which implies that it does not contain any of its vertices. Nevertheless, the ball contains the barycenters of the $(d-2)$-faces of Δ^d_d incident to x_i and therefore the entire near facet of Δ^d_{d-2}, as we now prove. Since Δ^d_{d-2} is a regular simplex, the distance between its barycenter and its vertices is R_{d-2}.

▶ **Lemma 9.** $R_d - I_d > R_{d-2}$.

Proof. Using (13) and (14), we simplify the expression for the difference on the left-hand side of the claimed inequality:

$$R_d - I_d = \sqrt{\frac{d}{d+1}} - \frac{1}{d(d+1)}\sqrt{\frac{d+1}{d}} = \frac{\sqrt{d+1}(d-1)}{\sqrt{d^3}}. \tag{15}$$

Dividing the claimed inequality by $R_{d-2} = \sqrt{(d-2)/(d-1)}$ and squaring, we get

$$\left[\frac{R_d - I_d}{R_{d-2}}\right]^2 = \left[\frac{\sqrt{d+1}(d-1)\sqrt{d-1}}{\sqrt{d^3}\sqrt{d-2}}\right]^2 = \frac{d^4 - 2d^3 + 2d - 1}{d^4 - 2d^3} > 1. \tag{16}$$

The claimed inequality follows. ◀

Finally note that the inradius of Δ^d_{d-1} is less than that of Δ^d_0: $I_d < J_d$. We can therefore choose $I_d < \varrho < \min\{J_d, R_d - R_{d-2}\}$, which is large enough so that $D(\varrho)$ is not contained in Δ^d_{d-1}, and it is small enough so that $D(\varrho)$ does not contain Δ^d_{d-1} and a touching Euclidean ball with center x_i contains the near facet of Δ^d_{d-2}.

7 Discussion

The contributions of this paper are geometric constraints on the location of the centers of smallest enclosing spheres for data in which dissimilarities are measured with Bregman divergences. The main tools used in their proofs are topological: the Nerve Theorem of Borsuk [2] and Leray [13] and the Fixed Point Lemma of Knaster, Kuratowski, and Mazurkiewicz [11]. Besides being of independent interest, the results are relevant to topological data analysis.

References

1 J.-D. Boissonnat, F. Nielsen and R. Nock. Bregman Voronoi diagrams. *Discrete Comput. Geom.* **44** (2010), 281–307.

2 K. Borsuk. On the imbedding of systems of compacta in simplicial complexes. *Fund. Math.* **35** (1948), 217–234.

3 L.M. Bregman. The relaxation method of finding the common point of convex sets and its applications to the solution of problems in convex programming. *USSR Comput. Math. Math. Phys.* **7** (1967), 200–217.

4 G.N. Clements. The geometry of phonological features. *Phonology Yearbook* **2** (1985), 225–252.

5 A. Dhara and J. Dutta. *Optimality Conditions in Convex Optimization: a Finite-dimensional View*. CRC Press, Taylor & Francis Group, Boca Raton, Florida, 2012.

6 H. Edelsbrunner, Ž. Virk and H. Wagner. Topological data analysis in information space. Manuscript, IST Austria, Klosterneuburg, Austria, 2017.

7 H. Edelsbrunner and H. Wagner. Topological data analysis with Bregman divergences. *In* "Proc. 33rd Ann. Sympos. Comput. Geom., 2017", 39:1–39:16.

8 J. Goldberger, S. Gordon and H. Greenspan. An efficient image similarity measure based on approximations of KL-divergence between two Gaussian mixtures. *In* "Proc. 9th IEEE Internat. Conf. Comput. Vision, 2003", 487.

9 A. Huang. Similarity measures for text document clustering. *In* "Proc. 6th New Zealand Comput. Sci. Research Student Conf., 2008", 49–56.

10 F. Itakura and S. Saito. An analysis-synthesis telephony based on the maximum likelihood method. *In* "Proc. 6th Internat. Congress Acoustics, 1968", Tokyo, Japan, c17–c20.

11 B. Knaster, C. Kuratowski and S. Mazurkiewicz. Ein Beweis des Fixpunktsatzes für n-dimensionale Simplexe. *Fund. Math.* **14** (1929), 132–137.

12 S. Kullback and R.A. Leibler. On information and sufficiency. *Ann. Math. Stat.* **22** (1951), 79–86.

13 J. Leray. Sur la forme des espaces topologiques et sur les points fixes des représentations. *J. Math. Pure Appl.* **24** (1945), 95–167.

14 F. Nielsen. An information-geometric characterization of Chernoff information. *IEEE Signal Process. Lett.* **20** (2013), 269–272.

15 F. Nielsen and R. Nock. On the smallest enclosing information disk. *In* "Proc. 18th Canad. Conf. Comput. Geom., 2006".

16 R. Nock, and F. Nielsen. Fitting the smallest enclosing Bregman ball. *In* "Proc. 16th European Conf. Machine Learning, 2005", 649–656.

17 A. Rind, T.D. Wang, W. Aigner, S. Miksch, K. Wongsuphasawat, C. Plaisant and B. Shneiderman. Interactive information visualization to explore and query electronic health records. *Found. Trends in Human-Comput. Interaction* **5** (2011), 207–298.

18 R.T. Rockafellar. *Convex Analysis*. Princeton Univ. Press, Princeton, New Jersey, 1970.

Near Isometric Terminal Embeddings for Doubling Metrics

Michael Elkin[1]
Department of Computer Science, Ben-Gurion University of the Negev
Beer-Sheva, Israel
elkinm@cs.bgu.ac.il

Ofer Neiman[2]
Department of Computer Science, Ben-Gurion University of the Negev
Beer-Sheva, Israel
neimano@cs.bgu.ac.il

Abstract

Given a metric space (X, d), a set of terminals $K \subseteq X$, and a parameter $t \geq 1$, we consider metric structures (e.g., spanners, distance oracles, embedding into normed spaces) that preserve distances for all pairs in $K \times X$ up to a factor of t, and have small size (e.g. number of edges for spanners, dimension for embeddings). While such terminal (aka source-wise) metric structures are known to exist in several settings, no terminal spanner or embedding with distortion close to 1, i.e., $t = 1 + \epsilon$ for some small $0 < \epsilon < 1$, is currently known.

Here we devise such terminal metric structures for *doubling* metrics, and show that essentially any metric structure with distortion $1 + \epsilon$ and size $s(|X|)$ has its terminal counterpart, with distortion $1 + O(\epsilon)$ and size $s(|K|) + 1$. In particular, for any doubling metric on n points, a set of $k = o(n)$ terminals, and constant $0 < \epsilon < 1$, there exists

- A spanner with stretch $1 + \epsilon$ for pairs in $K \times X$, with $n + o(n)$ edges.
- A labeling scheme with stretch $1 + \epsilon$ for pairs in $K \times X$, with label size $\approx \log k$.
- An embedding into ℓ_∞^d with distortion $1 + \epsilon$ for pairs in $K \times X$, where $d = O(\log k)$.

Moreover, surprisingly, the last two results apply if only K is a doubling metric, while X can be arbitrary.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms, Theory of computation → Sparsification and spanners

Keywords and phrases metric embedding, spanners, doubling metrics

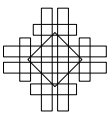
Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.36

Acknowledgements We are grateful to Paz Carmi for fruitful discussions.

1 Introduction

The area of *low-distortion embeddings* studies how well different metric spaces can be approximated by simpler, or more structured, metric spaces. Fundamental results in this realm include Bourgain's and Matousek's embeddings of general metrics into high-dimensional Euclidean and ℓ_∞ spaces [5, 24], respectively, Gupta et al.'s [19] embeddings of doubling metrics into normed spaces, and constructions of distance oracles and spanners for doubling

[1] This research was supported by the ISF grant No. (724/15).
[2] Supported in part by the ISF grant 1817/17 and BSF grant 2015813.



metrics [20, 16]. Linial et al. [23] and Bartal [4] demonstrated that low-distortion embeddings have numerous applications in Theoretical Computer Science.

All these embeddings [5, 24, 19] have inherent unavoidable dependencies in the total number of points n in both the distortion and in the dimension of the target space. In scenarios in which we have a metric space (X, d), and a subset $K \subseteq X$ of important points, aka *terminals*, the current authors and Filtser [14] demonstrated that one can devise *terminal embeddings*, i.e., embeddings that provide guarantees on the distortion of all pairs that involve a terminal in K, and whose guarantees on the distortion and the dimension depend on $k = |K|$, as opposed to the dependencies on n in the classical embeddings. Specifically, it is shown in [14] that essentially any known metric embedding into a normed space can be transformed via a general transformation into a terminal embedding, while incurring only a constant overhead in distortion.

This constant overhead does not constitute a problem when the distortion of the original embedding is $O(\log n)$, as is the case for Bourgain's embedding. However, for the important family of embeddings of *doubling* metrics [3, 19] the distortion in some cases is just $1 + \epsilon$, for an arbitrarily small $\epsilon > 0$. (The dimension grows with $1/\epsilon$.) This is also the case in the constructions of spanners and distance oracles for these metrics, due to [32, 16, 20]. Using the general transformation of [14] on them results in stretch c, for some constant $c \geq 1 + \sqrt{2}$, making the resulting embeddings and spanners far less appealing.

A metric (X, d) has doubling constant λ if any ball of radius $2R$ in the metric (for any $R > 0$) can be covered by at most λ radius-R balls. The parameter $\log_2 \lambda$ is called also the *doubling dimension* of the metric (X, d). A family of metrics is called *doubling* if the doubling dimension of each family member is constant.

Doubling metrics constitute a useful far-reaching generalization of Euclidean low-dimensional metrics. They have been extensively studied, see [3, 19, 6, 20, 16, 7, 18, 8, 15, 17, 28] and the references therein. Interestingly, these studies of doubling metrics have often produced improved bounds for low-dimensional Euclidean metrics as well. This was the case, e.g., for dynamic spanners for doubling and low-dimensional Euclidean metrics [18], spanners with low diameter, degree and weight [15], and fault-tolerant spanners [8].

In the current paper we devise a suit of terminal embeddings and metric structures, such as spanners, distance oracles and distance labeling schemes (see Section 2 for definitions), for doubling metrics with distortion $1 + \epsilon$, for an arbitrarily small $\epsilon > 0$. In particular, Gupta et al. [19] devised an embedding of metrics with doubling constant λ into ℓ_∞ with distortion $1 + \epsilon$ and dimension $\log n \cdot \lambda^{\log 1/\epsilon + O(1)}$. Our terminal embedding of doubling metrics into ℓ_∞ has *the same distortion*, but the dimension is $\log k \cdot \lambda^{\log 1/\epsilon + O(1)}$, i.e., the dependency on n is replaced by (essentially) the same dependency on k.

Johnson and Lindenstrauss [21] showed that any Euclidean metric can be embedded into an $O(\frac{\log n}{\epsilon^2})$-dimensional Euclidean one, with distortion $1+\epsilon$. While we are not able to provide a general terminal counterpart of this fundamental result, we do so in the important special case of doubling metrics. Specifically, we show that an Euclidean (possibly high-dimensional[3]) point set with doubling constant λ admits a terminal embedding with distortion $1 + \epsilon$ into an Euclidean space with dimension $O((\log k + \log \lambda \cdot \log 1/\epsilon)/\epsilon^2)$.

Har-Peled and Mendel [20], following [32], and extending previous classical results about low-dimensional Euclidean spanners (see, e.g., [2, 9, 12, 27]), showed that for any n-point metric with doubling constant λ and $\epsilon > 0$, there exists a $(1 + \epsilon)$-spanner with $n \cdot \lambda^{O(\log 1/\epsilon)}$ edges. Note that when ϵ is very small, the coefficient of n may be pretty large even in

[3] By "high-dimensional" we mean here typically dimension $\log n$ or greater.

Euclidean two-dimensional space. We devise a terminal $(1+\epsilon)$-spanner for doubling metrics with $n+k\cdot\lambda^{O(\log 1/\epsilon)}$ edges. In other words, when the number of terminals k is much smaller than n, the number of edges is just $n+o(n)$, as opposed to n multiplied by a large constant. (Note, however, that the distortion that our spanner provides is for pairs in $K\times X$, as opposed to $X\times X$.) To the best of our knowledge, no such terminal spanners are known even for two-dimensional Euclidean point sets. We also provide analogous terminal counterparts of Har-Peled and Mendel's distance oracles [20], and Slivkins' distance labeling schemes [31].

In addition, we study the setting in which the set of terminals K induces a doubling metric, while the entire point set X is a general (as opposed to doubling) metric. Surprisingly, we show that our terminal distance labeling and also embedding of doubling metrics into ℓ_∞ apply in this far more general scenario as well, with the same stretch $1+\epsilon$, and the same size/dimension as when X is a doubling metric. We also devise terminal spanners and terminal distance oracles for this more general scenario that K is doubling, while X is a general metric.

Related work. There has been several works which devised metric structures for partial subsets. Already [10] considered distance preservers for a designated set of pairs. In [11, 29, 22] pairwise spanners for general metrics were studied, and in particular terminal spanners. Recently [1] introduced reachability preservers from a given set of sources.

Interestingly, lately we realized that the general transformation from [14] can also be easily extended to produce terminal embeddings that apply to this general scenario (that points of $X\setminus K$ lie in a general metric, while points of K lie in a special metric). However, as was mentioned above, that transformation increases the stretch by at least a constant factor, and is thus incapable of producing terminal embeddings with stretch $1+\epsilon$.

The only known to us terminal metric structure with distortion $1+\epsilon$ is a prioritized distance labeling scheme for graphs that exclude a fixed minor, due to the current authors and Filtser [13]. In the current paper we provide the first near-isometric (i.e., having stretch $1+\epsilon$) terminal *spanners and embeddings*.

1.1 Technical overview

The naive approach for building a terminal spanner for a given metric space (X,d), is to apply a known construction on the set of terminals K, and extend the spanner to $X\setminus K$ by adding an edge from each point in $X\setminus K$ to its nearest terminal. (The same approach can be used for distance oracles/labeling and embeddings.) This is essentially the approach taken by [14] (albeit in a much more general setting). Unfortunately, such a construction cannot provide small $1+\epsilon$ stretch (it can be easily checked that it may give stretch at least 3). We need several ideas in order to provide small stretch.

First, we use the well known property of doubling metrics, that balls contain bounded size nets (see Section 2 for definitions). We construct nets in all relevant distance scales, and enrich K by a set $Y\supseteq K$ of net points. The points of Y are those net points that are, to a certain extent, close to K, depending on their distance scale. Then we apply a black-box construction of a spanner on the set Y. Finally, we extend the spanner to every $x\in X\setminus Y$, by adding a single edge from x : either to the nearest terminal, or to a single net point $y\in Y$. The set Y is carefully chosen so that each non-terminal $x\in X\setminus K$, either has a close-by terminal that "takes care" of it, and otherwise there is a net point $y\in Y$ sufficiently close to x so that x will have good stretch going via y.

One issue to notice is that even though Y is larger than K, it is still $|Y|=O(|K|)$ (at least for constant ϵ,λ). So we can have many points in $X\setminus Y$ that do not have a representative

$y \in Y$. The main technical part of the paper is devoted to proving that the particular choice of Y guarantees low stretch for any pair $(x, v) \in X \times K$, even when x has no representative $y \in Y$, by using the path through the nearest terminal to x.

It is instrumental to think of the set Y as an *"enriched"* terminal set. This idea of enriching the terminal set K with additional points may be useful in other settings as well.

In the setting when only K is doubling, our construction of terminal spanners (and also distance oracles/labeling schemes) is done by adding *multiple edges* from each $x \in X \setminus K$ to nearby terminals that constitute a net. This approach can not work, however, for embeddings into normed spaces. A certain type of embedding (such as the embedding of doubling metrics into ℓ_∞) can be used in a non-black-box manner, and we show how to incorporate the points of $X \setminus K$ into the embedding for K, without increasing the dimension.

2 Preliminaries

2.1 Embeddings, spanners and distance oracles/labeling scheme

Let (X, d) be a finite metric space. For a target metric (Z, d_Z), an *embedding* is a map $f : X \to Z$, and the *distortion* of f is the minimal α (in fact, it is the infimum), such that there exists a constant c that for all $x, y \in X$

$$d(x, y) \leq c \cdot d_Z(x, y) \leq \alpha \cdot d(x, y) \ . \tag{1}$$

When Z is the shortest path metric of a graph H and $c = 1$, we say that H is an *α-spanner* of (X, d). Given a set of terminals $K \subseteq X$, a *terminal* embedding guarantees (1) only for pairs in $K \times X$.

An *approximate distance oracle* is a data structure that can report a multiplicative approximation of $d(x, y)$, for all $x, y \in X$. For $K \subseteq X$, it is a *terminal* distance oracle if it can report only pairs in $K \times X$. The relevant parameters of an oracle are: its size (we measure the size in machine words), query time, and stretch factor (and to some extent, also the preprocessing time required to compute it). If one can distribute the data structure by storing a short label $L(x)$ at each vertex $x \in X$, and compute the approximation to $d(x, y)$ from $L(x)$ and $L(y)$ alone, this is called a *distance labeling scheme*.

For $x \in X$ and $r > 0$, let $B(x, r) = \{y \in X \ : \ d(x, y) \leq r\}$ be a closed ball. The doubling constant of X, denoted λ, is the minimal integer such that for every $r > 0$, every ball of radius $2r$ can be covered by λ balls of radius r.

2.2 Terminal nets

For $r > 0$, an *r-net* is a set $N \subseteq X$ satisfying the following:

1. For all $u, v \in N$, $d(u, v) > r$, and
2. for each $x \in X$, there exists $u \in N$ with $d(x, u) \leq r$.

The following claim is obtained by iteratively applying the definition of doubling constant.

▶ **Claim 1** ([19]). *Fix any $q, r > 0$, and let N be an r-net. For any $x \in X$ we have that*

$$|B(x, q) \cap N| \leq \lambda^{\log\lceil 2q/r \rceil} \ .$$

It is well-known that a greedy algorithm that iteratively picks an arbitrary point $u \in X$ to be in N, and removes every point within distance r of u, will create an r-net. Given a set of terminals $K \subseteq X$, we say that the greedy algorithm constructs a *terminal r-net*, if it prefers to take points from K until it is exhausted, and only then picks other points to N. We also

observe that given a terminal $2r$-net N, one may choose a terminal r-net N' that contains every *terminal* of N (by greedily picking to N' the terminals of N first – note that N' is not guaranteed to contain all points of N, just the terminals).

2.3 Extendable metric structure

Given a metric (X, d), we denote by $\hat{d}$ the distance function of some metric structure on it. We say that a family of structures is *extendable*, if the structure on a subset $Y \subseteq X$ can be extended to the entire X (so that $\hat{d}$ remains the same for pairs in Y), by *hanging* each $x \in X \setminus Y$ on some $u = u(x) \in Y$ and having that:

1. $\hat{d}(x, u) = d(x, u)$.
2. For any $v \in Y$, $\max\{d(x, u), \hat{d}(u, v)\} \leq \hat{d}(x, v) \leq d(x, u) + \hat{d}(u, v)$.

We argue that essentially all known structures are extendable. For each $x \in X \setminus Y$, let $u = u(x) \in Y$ be the point onto which x is hanged.

- **Spanners**. If the structure is a spanner on Y, then the extension for each x is done by adding the edge $\{x, u\}$ with weight $d(x, u)$. For any $v \in Y$, we indeed have that $\hat{d}(x, v) = d(x, u) + \hat{d}(u, v)$, satisfying both requirements.
- **Distance labeling**. For a distance labeling (or oracle), x stores the label of u and also $d(x, u)$. For a query on (x, v) where $v \in Y$, return $\hat{d}(x, v) = d(x, u) + \hat{d}(u, v)$.
- **Embeddings**. If the structure is an embedding $f : Y \to \ell_p^s$, then the extension $\hat{f}$ can be done by adding a new coordinate, and defining $\hat{f} : X \to \ell_p^{s+1}$ by setting for $v \in Y$, $\hat{f}(v) = (f(v), 0)$ and $f(x) = (f(u), d(x, u))$. Then we get that for all $v \in Y$, $\hat{d}(x, v) = \left(\hat{d}(u, v)^p + d(x, u)^p\right)^{1/p}$, which satisfies both requirements for every $1 \leq p \leq \infty$.

3 Terminal metric structures for doubling metrics

In this section we present our main result. For ease of notation, we measure the size of the structure as the size per point (e.g. for a spanner with m edges over n points we say the size is m/n). Our main result is:

▶ **Theorem 2.** *Let (X, d) be a metric space with $|X| = n$ that has doubling constant λ, and fix any set $K \subseteq X$ of size $|K| = k$. For $0 < \epsilon < 1$, assume that there exists an extendable metric structure for any $Y \subseteq X$ that has stretch $1 + \epsilon$ and size $s(|Y|)$, then there exists a structure for X with $1 + O(\epsilon)$ stretch for pairs in $K \times X$ and size $s(k \cdot \lambda^{O(\log(1/\epsilon))}) + 1$.*

The following corollary follows by applying this theorem with known embeddings/distance oracles/spanners constructions.

▶ **Corollary 3.** *Let (X, d) be a metric space with $|X| = n$ that has doubling constant λ, and fix any set $K \subseteq X$ of size $|K| = k$. Then for any $0 < \epsilon < 1$, the following metric structures exists:*

1. *If (X, d) is Euclidean, then there exists a terminal embedding into ℓ_2 with distortion $1 + \epsilon$ and dimension $O((\log k + \log \lambda \cdot \log(1/\epsilon))/\epsilon^2)$.*
2. *A terminal embedding into ℓ_∞ with distortion $1 + \epsilon$ and dimension $\log k \cdot \lambda^{\log(1/\epsilon)+O(1)} \cdot \log(1/\epsilon)$.*
3. *A terminal spanner for (X, d) with stretch $1 + \epsilon$ and $k \cdot \lambda^{O(\log(1/\epsilon))} + n$ edges.*
4. *A terminal distance oracle with stretch $1 + \epsilon$, with size $k \cdot \lambda^{O(\log(1/\epsilon))} + O(n)$ and query time $\lambda^{O(1)}$.*

5. *A terminal distance labeling scheme with stretch* $1+\epsilon$*, with label size* $\lambda^{O(\log(1/\epsilon))} \cdot \log k \cdot \log\log \Delta_k$ *(where* Δ_k *is the aspect ratio of* K*).*
6. *A terminal embedding into a distribution of tree-width* t *graphs*[4] *with expected distortion* $1+\epsilon$ *for* $t \leq \lambda^{O(\log\log\lambda + \log(1/\epsilon) + \log\log\Delta_K)}$.

Proof. The first item follows from [21], the second using [19, 28], the third and fourth items use [20] results, the fifth applies a result of [31], and the sixth from [32].[5] ◀

In what follows we prove Theorem 2. Let (X,d) be a metric space with $|X| = n$ and doubling constant λ, and let $K \subseteq X$ be a set of terminals. Fix any $0 < \epsilon < 1/20$, set $b = \lceil \log(1/\epsilon) \rceil$, and let $\Delta = \max_{u,v\in K}\{d(u,v)\}$, $\delta = \min_{u\neq v\in K}\{d(u,v)\}$ and $s = \lceil \log(\Delta/(\epsilon^2\delta)) \rceil$. Let $S = \{0,1,\ldots,s\}$, and for each $i \in S$ define $r_i = 2^i \cdot \epsilon^2\delta$. Observe that $r_0 = \epsilon^2\delta$ and $r_s \geq \Delta$.

3.1 Construction

3.1.1 Multi-scale partial partitions

We begin by constructing partial partitions, based on terminal nets, in various scales. The clusters of the partition at level i are created by iteratively taking balls of radius r_i centered at the points of a terminal r_i-net. Some of these balls may be sufficiently far away from K, we call such clusters *final*, and do not partition them in lower levels. See Algorithm 1 for the full details.

Algorithm 1 `Partial-Partitions` $((X,d),K)$

$R_s = X$;
for $i = s, s-1, \ldots, 0$ **do**
  Let $N_i = \{x_{i,1}, \ldots, x_{i,b_i}\}$ be a terminal r_i-net of R_i; (For $i < s$, each $u \in K \cap N_{i+1}$ will be in N_i as well);
  for $j = 1, \ldots, b_i$ **do**
    $C_{i,j} \leftarrow B(x_{i,j}, r_i) \cap R_i$;
    $R_i \leftarrow R_i \setminus C_{i,j}$;
    if $d(x_{i,j}, K) \geq r_i/\epsilon$ **then**
      Let final$(C_{i,j}) = true$;
    else
      Let final$(C_{i,j}) = false$
    end if
  end for
  $R_{i-1} = \bigcup_{j \,:\, \text{final}(C_{i,j})=false} C_{i,j}$;
end for

For every scale $i \in S$ this indeed forms a partition of $R_i \subseteq X$, because N_i is an r_i-net. Also, every cluster $C_{i,j}$ in the partition of R_i has a center $x_{i,j}$. Observe that every cluster containing a terminal is not final, and that each point in X has at most one final cluster containing it. In addition, the definition of terminal net guarantees that the prefix of N_i consists of terminals, so each terminal $u \in K$ must be assigned to a cluster centered at a terminal. Finally, notice that at level 0, every terminal is a center of its own cluster (since $r_0 < \delta$).

[4] See [30] for definition of tree-width.
[5] For the last two results, we note that our proof provides $Y \supseteq K$ satisfying $\Delta_Y \leq O(\Delta_K/\epsilon^4)$, on which we apply the labeling scheme of [31], or the embedding of [32].

3.1.2 Marking stage

We now mark some of the clusters, these marked clusters are the "important" clusters whose center will participate in the black-box construction. For every terminal $u \in K$, let $i_u \in S$ be the maximal index such that $u \in N_{i_u}$, and mark every cluster $C_{i,j}$ with center $x_{i,j}$ satisfying both conditions (recall that $b = \lceil \log(1/\epsilon) \rceil$.)

1. $i_u - 2b \leq i \leq i_u$, and
2. $d(u, x_{i,j}) \leq 2r_{i_u}/\epsilon^2$.

3.1.3 Constructing the metric structure

Let $Y \subseteq X$ be the collection of centers of marked clusters (note that $K \subseteq Y$). Apply the black-box construction on Y, and extend it to $X \setminus Y$ as follows. For every $x \in X$ that lies in a final marked cluster C with center y, hang x on y (recall that x can be in at most one final cluster). In every other case (e.g., x is in a final unmarked cluster, or does not have a final cluster containing it), hang x on $u \in K$, the nearest terminal to x.

3.2 Analysis

First we show that $|Y|$ is sufficiently small.

▶ **Claim 4.** $|Y| \leq |K| \cdot \lambda^{5b}$.

Proof. We will show that each $u \in K$ marks at most λ^{5b} clusters. By Claim 1, the ball $B(u, r_{i_u+2b+1})$ contains at most $\lambda^{\log(r_{i_u+2b+1}/r_{i_u-2b})} = \lambda^{4b+2}$ net points of N_{i_u-2b} (and only less net points from the other nets $N_{i_u-2b+1}, \ldots, N_{i_u}$). The second condition for marking implies that only centers in this ball can be marked by u. Since there are $2b+1$ possible levels $i \in [i_u - 2b, i_u]$, at most $(2b+1) \cdot \lambda^{4b+2} \leq \lambda^{5b}$ clusters may be marked by u. ◀

The bound on the size follows from Claim 4, and from the fact that each point in $X \setminus Y$ is hanged from a single $y \in Y$, so it requires a single edge/memory word/coordinate. It remains to bound the stretch by $1 + O(\epsilon)$ for pairs in $K \times X$. By the assumption, the metric structure for Y induces a distance function $\hat{d}$ which is a $1+\epsilon$ approximation of d, w.l.o.g we assume that distances cannot contract, and expand by a factor of at most $1+\epsilon$. Fix some $x \in X$ and $v \in K$. Recall that by definition, if x was hanged on $u \in Y$, then $\hat{d}(x, u)$ must satisfy

$$\max\{d(x,u), \hat{d}(u,v)\} \leq \hat{d}(x,u) \leq d(x,u) + \hat{d}(u,v) \ .$$

Consider the following cases.

Case 1: x does not have a final cluster containing it. In this case x lies very close to its nearest terminal $u \in K$, and all other terminals are at least $1/\epsilon$ times farther away, so the stretch guaranteed for u will suffice for x. More formally: the cluster C containing x at level 0 centered at y is not final, that is, $d(y, K) < r_0/\epsilon$. Since C has radius $r_0 = \epsilon^2\delta$, we have that

$$d(x,u) = d(x,K) \leq d(x,y) + d(y,K) \leq \epsilon^2\delta + \epsilon\delta = (1+\epsilon)\epsilon\delta \ . \tag{2}$$

We have that $d(u,v) \leq d(u,x) + d(x,v) \leq (1+\epsilon)\epsilon\delta + d(x,v) \leq 2\epsilon \cdot d(u,v) + d(x,v)$, so that

$$d(u,v) \leq d(x,v)/(1-2\epsilon) \ . \tag{3}$$

Since $\hat{d}$ approximates d with stretch $1+\epsilon$ on K,

$$\begin{aligned}\hat{d}(x,v) &\leq d(x,u)+\hat{d}(u,v)\\ &\leq d(x,u)+(1+\epsilon)d(u,v)\\ &\overset{(2)}{\leq} (1+\epsilon)\epsilon\delta+(1+\epsilon)d(u,v)\leq(1+3\epsilon)d(u,v)\\ &\overset{(3)}{\leq} (1+6\epsilon)d(x,v)\ ,\end{aligned}$$

where the last two inequalities use that $\epsilon < 1/12$. On the other hand,

$$\begin{aligned}\hat{d}(x,v) &\geq \hat{d}(u,v)\\ &\geq d(u,v)\\ &= (1-\epsilon)\cdot d(u,v)+\epsilon\cdot d(u,v)\\ &\geq (1-\epsilon)\cdot(d(x,v)-d(x,u))+\epsilon\delta\\ &\overset{(2)}{\geq} (1-\epsilon)\cdot d(x,v)-(1-\epsilon)(1+\epsilon)\epsilon\delta+\epsilon\delta\\ &\geq (1-\epsilon)\cdot d(x,v)\ .\end{aligned}$$

Case 2: x lies in a final *marked* cluster. Let C be the final marked cluster at level $i\in S$ with center y that contains x. In this case we show that $d(x,y)$ is smaller by roughly $1/\epsilon$ than $d(x,K)$, so that the stretch guaranteed for $y\in Y$ will also be sufficient for x. Since C is final, $d(y,v)\geq d(y,K)>r_i/\epsilon$, therefore

$$d(x,v)\geq d(y,v)-d(x,y)\geq r_i/\epsilon-r_i>r_i/(2\epsilon)\ . \tag{4}$$

Using that the structure built for Y has stretch at most $1+\epsilon$, we have that

$$\begin{aligned}\hat{d}(x,v) &\leq d(x,y)+\hat{d}(y,v)\\ &\leq d(x,y)+(1+\epsilon)d(y,v)\\ &\leq d(x,y)+(1+\epsilon)(d(x,y)+d(x,v))\\ &= (2+\epsilon)d(x,y)+(1+\epsilon)d(x,v)\\ &\leq (2+\epsilon)r_i+(1+\epsilon)d(x,v)\\ &\overset{(4)}{\leq} 2\epsilon(2+\epsilon)d(x,v)+(1+\epsilon)d(x,v)\\ &\leq (1+6\epsilon)d(x,v)\ .\end{aligned}$$

And also,

$$\begin{aligned}\hat{d}(x,v) &\geq \hat{d}(y,v)\\ &\geq d(y,v)\\ &\geq d(x,v)-d(x,y)\\ &\geq d(x,v)-r_i\\ &\overset{(4)}{\geq} (1-2\epsilon)d(x,v)\ .\end{aligned}$$

Case 3: x lies in a final *non-marked* cluster C. Let u be the nearest terminal to x. Intuitively, since x is in a final cluster, all terminals are $1/\epsilon$ farther away than the radius of C. However, since C is not marked, its center does not participate in the black-box construction for Y. Fortunately, the marking of clusters guarantees that u, the closest terminal to x, must be in a terminal net of very high scale (otherwise it would have marked C), and it follows that every other terminal is either very far away from u (and thus from x as well), or very close to u. Surprisingly, in both cases we can use the stretch bound guaranteed for K. We prove this observation formally in the following lemma.

▶ **Lemma 5.** *For any point x contained in a final non-marked cluster C of level i with $i < s$, there exists a terminal $u' \in K$ such that $d(x,u') \in [r_i/(2\epsilon), 3r_i/\epsilon]$ and for any other terminal $w \in K$ it holds that $d(u',w) \leq r_i$ or $d(u',w) \geq r_i/\epsilon^2$.*

Proof. Since C with center y is the only final cluster containing x, the cluster C' with center y' containing x at level $i+1$ is not final (recall we assume $i < s$). Thus there exists a terminal $z \in K$ with $d(y',z) \leq r_{i+1}/\epsilon$. Consider the terminal $u' \in N_{i+1}$ which is the center of the cluster containing z at level $i+1$ (we noted above that clusters containing a terminal must have a terminal as a center). By the triangle inequality $d(x,u') \leq d(x,y') + d(y',z) + d(z,u') \leq r_{i+1} + r_{i+1}/\epsilon + r_{i+1} < 3r_i/\epsilon$ (note that the same bound holds for $d(y,u')$). On the other hand, since C is final we have that $d(y,u') \geq r_i/\epsilon$, and thus $d(x,u') \geq d(y,u') - d(y,x) \geq r_i/\epsilon - r_i \geq r_i/(2\epsilon)$.
Next we show that $u' \in N_{i+2b+1}$. Seeking contradiction, assume $u' \notin N_{i+2b+1}$ (or that $i \geq s-2b$ so such a net does not exist), and consider the largest j such that $u' \in N_j$. Since the nets are hierarchical and $u' \in N_{i+1}$, it must be that $i+1 \leq j \leq i+2b$, which implies that $d(u',y) \leq 3r_i/\epsilon < r_{i+b+2} < 2r_j/\epsilon^2$. By the marking procedure, the cluster C would have been marked by u'. Contradiction. We conclude that $u' \in N_{i+2b+1}$.
Fix any terminal $w \in K$, and we know show that $d(u',w) \leq r_i$ or $d(u',w) \geq r_i/\epsilon^2$. Seeking contradiction, assume that $r_i < d(u',w) < r_i/\epsilon^2$. Let $v' \in K$ be the center of the cluster containing w at level i, that is $v' \in N_i$. Note that $d(v',w) \leq r_i$, and thus $v' \neq u'$. Since N_{i+2b+1} is an $r_{i+2b+1} = 2r_i/\epsilon^2$ net, and as $d(u',v') \leq r_i + r_{i+2b}$, it must be that $v' \notin N_{i+2b+1}$. The contradiction will follow once we establish that v' will mark C. Indeed, the largest j such that $v' \in N_j$ satisfies $i \leq j \leq i+2b$, and also $d(v',y) \leq d(v',w) + d(w,u') + d(u',y) \leq r_i + r_{i+2b} + 3r_{i+b} \leq 2r_j/\epsilon^2$, so C should have been marked. ◀

Next, we prove the stretch bound for the pair (x,v). Observe that if the final cluster C containing x and centered at y is of level s, then $d(y,K) \geq r_s/\epsilon$, and thus

$$d(x,K) \geq d(y,K) - d(y,x) \geq r_s/(2\epsilon)\ . \tag{5}$$

This implies that

$$\begin{aligned}
\hat{d}(x,v) &\leq d(x,u) + \hat{d}(u,v) \\
&\leq d(x,u) + (1+\epsilon)d(u,v) \\
&\leq d(x,v) + (1+\epsilon)r_s \\
&\overset{(5)}{\leq} d(x,v) + 2\epsilon(1+\epsilon)d(x,v) \\
&\leq (1+3\epsilon)d(x,v)\ .
\end{aligned}$$

Since $d(u,v) \leq \Delta \leq r_s$, we get that

$$\begin{aligned}
\hat{d}(x,v) &\geq d(x,u) \\
&\geq (1-2\epsilon)\cdot(d(x,v) - d(u,v)) + 2\epsilon \cdot d(x,u) \\
&\overset{(5)}{\geq} (1-2\epsilon)\cdot d(x,v) - r_s + r_s \\
&\geq (1-2\epsilon)\cdot d(x,v)\ .
\end{aligned}$$

From now on we may assume that C is of level i with $i < s$. By Lemma 5 there exists $u' \in K$ such that $d(x,u') \in [r_i/(2\epsilon), 3r_i/\epsilon]$ and for any terminal $w \in K$, it holds that $d(u',w) \leq r_i$ or $d(u',w) \geq r_i/\epsilon^2$. Note that since u is the nearest terminal to x, it must be that $d(u,u') \leq r_i$, so we have that $d(x,u) \in [r_i/(3\epsilon), 4r_i/\epsilon]$. Finally, we consider the

two cases for v: close or far from u'.

Sub-case a: $d(u',v) \le r_i$. In this case $d(u,v) \le 2r_i$, and thus $d(x,v) \ge d(x,u) - d(u,v) \ge r_i/(3\epsilon) - 2r_i \ge r_i/(4\epsilon)$. It follows that

$$\begin{aligned}\hat{d}(x,v) &\le d(x,u) + \hat{d}(u,v)\\ &\le d(x,u) + (1+\epsilon)d(u,v)\\ &\le d(x,v) + (1+\epsilon)2r_i\\ &\le d(x,v) + 5r_i\\ &\le (1+9\epsilon)d(x,v)\ .\end{aligned}$$

Since $d(u,v) \le 2r_i \le 8\epsilon \cdot d(x,v)$, we also have

$$\begin{aligned}\hat{d}(x,v) &\ge d(x,u)\\ &\ge d(x,v) - d(u,v)\\ &\ge (1-8\epsilon)\cdot d(x,v)\ .\end{aligned}$$

Sub-case b: $d(u',v) \ge r_i/\epsilon^2$. Now we have that $d(u',v) \le d(u',x) + d(x,v) \le 3r_i/\epsilon + d(x,v) \le 3\epsilon d(u',v) + d(x,v)$, and so $d(u',v) \le d(x,v)/(1-3\epsilon)$. It follows that

$$\begin{aligned}\hat{d}(x,v) &\le d(x,u) + \hat{d}(u,v)\\ &\le d(x,u) + (1+\epsilon)d(u,v)\\ &\le (2+\epsilon)d(x,u) + (1+\epsilon)d(x,v)\\ &\le (2+\epsilon)4r_i/\epsilon + (1+\epsilon)d(x,v)\\ &\le 9\epsilon\cdot d(u',v) + (1+\epsilon)d(x,v)\\ &\le (1+12\epsilon)d(x,v)\ .\end{aligned}$$

Using that $d(u,u') \le r_i$ and that $d(x,v) \ge (1-3\epsilon)d(u',v) \ge (1-3\epsilon)r_i/\epsilon^2 \ge r_i/(2\epsilon^2)$, we conclude that

$$\begin{aligned}\hat{d}(x,v) &\ge \hat{d}(u,v)\\ &\ge d(u,v) \ge d(v,x) - d(x,u') - d(u',u)\\ &\ge d(v,x) - 3r_i/\epsilon - r_i\\ &\ge (1-8\epsilon)\cdot d(v,x) + 8\epsilon\cdot r_i/(2\epsilon^2) - 3r_i/\epsilon - r_i\\ &\ge (1-8\epsilon)\cdot d(v,x)\ .\end{aligned}$$

4 The case where only K is doubling

So far we assumed that the entire metric (X,d) is doubling. It is quite intriguing to understand what results can be obtained where only the terminal set K is doubling, while X is arbitrary. We show that in such a case one can obtain terminal metric structures with guarantees similar to the standard results (non-terminal) that apply when the entire metric (X,d) is doubling. For spanners and distance labeling this follow by a simple extension of the black-box result, but unlike [26, 14], we use multiple points of K for extending each $x \in X \setminus K$.

▶ **Theorem 6.** *Let (X,d) be a metric space on n points, and let $K \subseteq X$ so that (K,d) has doubling constant λ. Then for any $0 < \epsilon < 1$ there exist:*

- *A terminal spanner with stretch $1+\epsilon$ and $O(n \cdot \lambda^{O(\log(1/\epsilon))})$ edges.*
- *A terminal distance oracle with stretch $1+\epsilon$, size $n \cdot \lambda^{O(\log(1/\epsilon))}$, and query time $\lambda^{O(1)}$.*
- *A terminal labeling scheme with stretch $1+\epsilon$, with label size $\lambda^{O(\log(1/\epsilon))} \log k \cdot \log\log \Delta_k$ (where Δ_k is the aspect ratio of K).*

Observe that the result for the labeling scheme seems to improves Corollary 3, which requires that the whole metric is doubling. (In fact, the label size in Theorem 6 is slightly larger, this fact is hidden by the constant in the $O(\cdot)$ notation.)

For embeddings, it is unclear how to use this extension approach, since it involves multiple points. We thus need to adjust the embedding itself. As an example to this adjustment, we have the following result, which strictly improves the corresponding item in Corollary 3. Its proof is in Section 4.2.

▶ **Theorem 7.** *Let (X,d) be a metric space, and let $K \subseteq X$ of size $|K| = k$ so that (K,d) has doubling constant λ. Then for any $0 < \epsilon < 1$ there exists a terminal embedding of X into ℓ_∞ with distortion $1+\epsilon$, and dimension $\log k \cdot \lambda^{O(\log(1/\epsilon))}$.*

We remark that any embedding of (X,d) into ℓ_∞ with distortion less than 3 for all pairs, requires in general dimension $\Omega(n)$ [25]. We also note that a terminal version of the JL lemma is impossible whenever only K is Euclidean, and $X \setminus K$ is not. To see this, note that any three vertices of $K_{2,2}$ admit an isometric embedding to ℓ_2, but embedding all four requires distortion $\sqrt{2}$. When only one vertex is non-terminal, all pairwise distances must be preserved up to $1+\epsilon$, which is impossible for $\epsilon < 1/3$, say.

4.1 Proof of Theorem 6

We prove the spanner result first. Let H be a spanner for (K,d) with stretch $1+\epsilon$ and $k \cdot \lambda^{O(\log(1/\epsilon))}$ edges given by [20], say. For any $x \in X$, let $u = u(x) \in K$ be the closest terminal to x, and denote $R = d(x,u)$. Take $N(x)$ to be an ϵR-net of $B(x, 2R/\epsilon) \cap K$, by Claim 1, $|N(x)| \leq \lambda^{O(\log(1/\epsilon))}$. Add the edges $\{(x,v)\}_{v \in N(x)}$, each with weight $d(x,v)$ to the spanner. Since we added $\lambda^{O(\log(1/\epsilon))}$ edges for each point, the bound on the number of edges follows, and it remains to bound the stretch by $1+O(\epsilon)$. Clearly no distances can contract, and we bound the expansion. Fix $x \in X$ and $v \in K$, and denote $u = u(x)$ with $R = d(x,u)$. In the case $v \notin B(x, 2R/\epsilon)$ we have that $R \leq \epsilon \cdot d(x,v)/2$, so that

$$\begin{aligned} d_H(x,v) &\leq d_H(x,u) + d_H(u,v) \leq d(x,u) + (1+\epsilon)d(u,v) \\ &\leq (2+\epsilon)d(x,u) + (1+\epsilon)d(x,v) = (2+\epsilon)R + (1+\epsilon)d(x,v) \\ &\leq (1+3\epsilon)d(x,v) \ . \end{aligned}$$

Otherwise, $v \in B(x, 2R/\epsilon)$. Let $v' \in N(x)$ be the nearest net point to v, with $d(v,v') \leq \epsilon R \leq \epsilon \cdot d(x,v)$ (recall u is the nearest terminal to x). Then

$$\begin{aligned} d_H(x,v) &\leq d_H(x,v') + d_H(v',v) \\ &\leq d(x,v') + (1+\epsilon)d(v',v) \\ &\leq d(x,v) + (2+\epsilon)d(v',v) \\ &\leq d(x,v) + (2+\epsilon)\epsilon \cdot d(x,v) \\ &\leq (1+3\epsilon)d(x,v) \ . \end{aligned}$$

The proof for the labeling scheme (and also distance oracle) is similar. Apply the black-box scheme on (K,d), and for each $x \in X \setminus K$ define $N(x)$ as above, and x stores all labels for $v' \in N(x)$ along with $d(x,v')$. Given a query (x,v), return $\min_{v' \in N(x)}\{d(x,v') + \hat{d}(v,v')\}$, where $\hat{d}$ is the distance function of the labeling scheme.

4.1.1 Lower bound

We now show that when only K is doubling, one cannot achieve a result as strong as Theorem 2 (there the number of edges in a spanner with stretch $1+\epsilon$ can be as low as $n + o(n)$). In fact, Theorem 6 is tight up to a constant factor in the exponent of λ.

▶ **Claim 8.** *There exists a constant $c > 0$, so that for any (sufficiently large) integer n and any integer $\lambda > 1$, there is a metric (X, d) on n points with a subset $K \subseteq X$, so that (K, d) has doubling constant $O(\lambda)$, but for any $0 < \epsilon < 1$, any terminal spanner of X with stretch $1 + \epsilon$ must have at least $n \cdot \lambda^{\log(c/\epsilon)}$ edges.*

Proof. Let $t = \lceil \log \lambda \rceil$, and let K be an ϵ-net of the unit sphere of $\mathbb{R}^t$. It is well known that $|K| = \Theta(1/\epsilon)^{t-1} = \lambda^{\log(c/\epsilon)}$ for some constant c.

Define (X, d) by setting for each $x, y \in X$, $d(x, y) = \begin{cases} \|x - y\|_2 & x, y \in K \\ 1 & x \in X \setminus K, y \in K \\ 2 & x, y \in X \setminus K \end{cases}$. Note that distances between points in K correspond to the Euclidean distance, and are at most 2, so that K has doubling constant $O(\lambda)$. Observe that any spanner with stretch $1 + \epsilon$ must contain all the edges in $K \times X$, because the distance between any two points in K is larger than ϵ, so any path from $x \in X \setminus K$ to $y \in K$ that does not contain the edge (x, y), will be of length greater than $1 + \epsilon$. ◀

4.2 Proof of Theorem 7

We follow the embedding technique of [28], but with different edge contractions defined below. Assume w.l.o.g that the minimal distance in (X, d) is 1. Let $\Delta = \mathrm{diam}(X)$, and for all $0 \le i \le \log \Delta$ let (X, d_i) be the metric defined as follows: consider the complete graph on vertex set X, with edge $\{u, v\}$ having weight $d(u, v)$. For every $x \in X$ and $v \in K$ with $d(x, v) < 2^{i-1} \cdot \epsilon/k$, replace the weight of this edge by 0, and let d_i be the shortest path metric on this graph. Since any shortest path in this graph has at most $2k$ edges that contain a vertex in K, we have that $d(x, y) - \epsilon \cdot 2^i \le d_i(x, y) \le d(x, y)$ for all $x, y \in X$.

For each $0 \le i \le \log \Delta$ take a r_i-net N_i with respect to (K, d_i) (i.e., take only terminals to the net), where $r_i = \epsilon \cdot 2^{i-2}$. Partition each N_i into $t = \lambda^{O(\log(1/\epsilon))}$ sets $N_{i1}, \ldots, N_{it}$, such that for each $u, v \in N_{ij}$, $d_i(u, v) \ge 5 \cdot 2^i$. (To obtain N_{ij}, one can greedily choose points from $N_i \setminus (\bigcup_{j' < j} N_{ij'})$ until no more can be chosen. See [28] for details.) Next we define the embedding, fix $D = \lceil 2t \log(2k/\epsilon) \rceil$, and let $\{e_0, \ldots, e_{D-1}\}$ be the standard orthonormal basis for $\mathbb{R}^D$, extended to an infinite sequence $\{e_j\}_{j \in \mathbb{N}}$ (that is, $e_j = e_{j \pmod D}$ for all $j \in \mathbb{N}$). For any $0 \le i \le \log \Delta$ and $0 \le j \le t - 1$, for $x \in X$ let

$$g_{ij}(x) = \min\{2^{i+1}, d_i(x, N_{ij})\} .$$

Define the embedding $f : X \to \mathbb{R}^D$ by

$$f(x) = \sum_{i=0}^{\log \Delta} \sum_{j=0}^{t-1} g_{ij}(x) \cdot e_{it+j} .$$

Expansion Bound. Now we show that the embedding f under the ℓ_∞ norm does not expand distances for pairs in $X \times K$ by more than a factor of $1 + \epsilon$. Fix a pair $x \in X$ and $v \in K$, and consider the h-th coordinate of the embedding f_h, with $0 \le h \le D - 1$. We have that $f_h(x) - f_h(v) = \sum_{i,j \,:\, h = it + j \pmod D} g_{ij}(x) - g_{ij}(v)$. Let $0 \le i' \le \log \Delta$ be such that $2^{i'-1} \le d(x, v) < 2^{i'}$, then for all $i > i' + \log(2k/\epsilon)$ it holds that $d(x, v) < 2^{i-1} \cdot \epsilon/k$ and thus $d_i(x, v) = 0$, in particular, $g_{ij}(x) = g_{ij}(v)$ and so there is no contribution at all from such scales. By the triangle inequality we also have that $g_{ij}(x) - g_{ij}(v) \le d_i(x, v)$ and

$g_{ij}(x) - g_{ij}(v) \leq 2^{i+1}$ for all $0 \leq i \leq \log \Delta$ and $0 \leq j \leq t-1$.

$$\begin{aligned} f_h(x) - f_h(v) &\leq \sum_{i,j \,:\, i \leq i' + \log(2k/\epsilon), h = it+j (\mathrm{mod}\ D)} g_{ij}(x) - g_{ij}(v) \\ &\leq \sum_{i,j \,:\, i' - \log(2k/\epsilon) < i \leq i' + \log(2k/\epsilon), h = it+j (\mathrm{mod}\ D)} g_{ij}(x) - g_{ij}(v) \\ &\quad + \sum_{i \leq i' - \log(2k/\epsilon)} 2^{i+1} \\ &\leq d_i(x,v) + 2^{i'+1} \cdot \epsilon/k \\ &\leq d(x,v)(1+\epsilon) \ . \end{aligned}$$

The third inequality holds, since by the choice of D there is at most one possible choice of i, j with $i' - \log(2k/\epsilon) < i < i' + \log(2k/\epsilon)$ such that $h = it + j (\mathrm{mod}\ D)$, and the last inequality uses that $k \geq 4$. By symmetry it follows that $|f_h(x) - f_h(v)| \leq d(x,v)(1+\epsilon)$, and thus $|f(x) - f(v)| \leq d(x,v)(1+\epsilon)$.

Contraction Bound. Now we bound the contraction of the embedding for pairs containing a terminal. Fix $x \in X$ and $v \in K$. We will show that there exists a single coordinate $0 \leq h \leq D-1$ such that $|f_h(x) - f_h(v)| \geq (1-\epsilon) d(x,v)$. Let $0 \leq i \leq \log \Delta$ such that $2^i \leq d(x,v) < 2^{i+1}$, and let $0 \leq j \leq t-1$ be such that $d_i(v, N_{ij}) \leq r_i$ (such a j must exist because N_i is an r_i-net of K). Denote by $u \in N_{ij}$ the point satisfying $d_i(v, N_{ij}) = d_i(v,u)$. Since $r_i = \epsilon \cdot 2^{i-2}$ also $g_{ij}(v) \leq r_i$.

We claim that $d_i(x, N_{ij}) = d_i(x,u)$. To see this, first observe that $d_i(x,u) \leq d_i(x,v) + d_i(v,u) \leq 2^{i+1} + r_i < (5/4) \cdot 2^{i+1}$. Consider any other $y \in N_{ij}$, by the construction of N_{ij}, $d_i(y,u) \geq 5 \cdot 2^i$, so $d_i(y,x) \geq d_i(y,u) - d_i(x,u) > (5/2) \cdot 2^{i+1} - (5/4) \cdot 2^{i+1} = (5/4) \cdot 2^{i+1} > d_i(x,u)$. Thus it follows that either $g_{ij}(x) = 2^{i+1} \geq d_i(x,y)$, or $g_{ij}(x) = d_i(x,u) \geq d_i(x,v) - d_i(v,u) \geq d_i(x,v) - r_i$. Using that $d_i(x,v) \geq d(x,v) - \epsilon \cdot 2^i$, we conclude that

$$g_{ij}(x) - g_{ij}(v) \geq (d_i(x,v) - r_i) - r_i = d_i(x,v) - \epsilon \cdot 2^{i-1} \geq d(x,v) - 2\epsilon \cdot 2^i \geq (1-2\epsilon) \cdot d(x,v) \ .$$

Let $0 \leq h \leq D-1$ be such that $h = it + j (\mathrm{mod}\ D)$, for the values of i, j fixed above. Then we claim that any other pair i', j such that $h = i'k + j (\mathrm{mod}\ D)$ has either 0 or very small contribution to the h coordinate. If $i' > i$ then it must be that $i' \geq \log(2k/\epsilon) + i + 1$ so that $d(x,v) \leq 2^{i+1} < 2^{i'-1} \cdot \epsilon/k$, thus as before $g_{i'j}(x) = g_{i'j}(v)$. For values of i' such that $i' < i$, then $i' \leq i - \log(2k/\epsilon)$, thus

$$\begin{aligned} \sum_{i' < i, j \,:\, h = i't + j (\mathrm{mod}\ D)} |g_{i'j}(x) - g_{i'j}(v)| &\leq \sum_{i' \leq i - \log(2k/\epsilon)} 2^{i'+1} \\ &\leq 2^i \cdot 2\epsilon/k \\ &\leq \epsilon \cdot d(x,v) \ . \end{aligned}$$

Finally,

$$\begin{aligned} \|f(x) - f(v)\|_\infty &\geq |f_h(x) - f_h(v)| \\ &\geq |g_{ij}(v) - g_{ij}(x)| - \sum_{i' < i, j \,:\, h = i't + j (\mathrm{mod}\ D)} |g_{i'j}(x) - g_{i'j}(v)| \\ &\geq d(x,v)(1-3\epsilon) \ . \end{aligned}$$

References

1. Amir Abboud and Greg Bodwin. Reachability preservers: New extremal bounds and approximation algorithms. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 1865–1883, 2018. doi:10.1137/1.9781611975031.122.
2. I. Althöfer, G. Das, D. Dobkin, D. Joseph, and J. Soares. On sparse spanners of weighted graphs. *Discrete Comput. Geom.*, 9:81–100, 1993.
3. P. Assouad. Plongements lipschitziens dans $\mathbb{R}^n$. *Bull. Soc. Math. France*, 111(4):429–448, 1983.
4. Y. Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *Proceedings of the 37th IEEE Symp. on Foundations of Computer Science*, pages 184–193, 1996.
5. J. Bourgain. On lipschitz embedding of finite metric spaces in hilbert space. *Israel Journal of Mathematics*, 52(1-2):46–52, 1985. doi:10.1007/BF02776078.
6. T-H. Hubert Chan and Anupam Gupta. Small hop-diameter sparse spanners for doubling metrics. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*, SODA '06, pages 70–78, Philadelphia, PA, USA, 2006. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=1109557.1109566.
7. T.-H. Hubert Chan, Anupam Gupta, Bruce M. Maggs, and Shuheng Zhou. On hierarchical routing in doubling metrics. *ACM Trans. Algorithms*, 12(4):55:1–55:22, aug 2016. doi:10.1145/2915183.
8. T.-H. Hubert Chan, Mingfei Li, Li Ning, and Shay Solomon. New doubling spanners: Better and simpler. *SIAM J. Comput.*, 44(1):37–53, 2015. doi:10.1137/130930984.
9. Barun Chandra, Gautam Das, Giri Narasimhan, and José Soares. New sparseness results on graph spanners. In *Proc. of 8th SOCG*, pages 192–201, 1992.
10. D. Coppersmith and M. Elkin. Sparse source-wise and pair-wise distance preservers. In *SODA: ACM-SIAM Symposium on Discrete Algorithms*, pages 660–669, 2005.
11. Marek Cygan, Fabrizio Grandoni, and Telikepalli Kavitha. On pairwise spanners. In *30th International Symposium on Theoretical Aspects of Computer Science, STACS 2013, February 27 - March 2, 2013, Kiel, Germany*, pages 209–220, 2013. doi:10.4230/LIPIcs.STACS.2013.209.
12. Gautam Das, Paul J. Heffernan, and Giri Narasimhan. Optimally sparse spanners in 3-dimensional euclidean space. In *Proceedings of the Ninth Annual Symposium on Computational GeometrySan Diego, CA, USA, May 19-21, 1993*, pages 53–62, 1993. doi:10.1145/160985.160998.
13. Michael Elkin, Arnold Filtser, and Ofer Neiman. Prioritized metric structures and embedding. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 489–498, 2015. doi:10.1145/2746539.2746623.
14. Michael Elkin, Arnold Filtser, and Ofer Neiman. Terminal embeddings. *Theor. Comput. Sci.*, 697:1–36, 2017. doi:10.1016/j.tcs.2017.06.021.
15. Michael Elkin and Shay Solomon. Optimal euclidean spanners: Really short, thin, and lanky. *J. ACM*, 62(5):35:1–35:45, 2015. doi:10.1145/2819008.
16. Jie Gao, Leonidas J. Guibas, and An Nguyen. Deformable spanners and applications. *Comput. Geom. Theory Appl.*, 35(1-2):2–19, 2006. doi:10.1016/j.comgeo.2005.10.001.
17. Lee-Ad Gottlieb. A light metric spanner. In *Proc. of 56th FOCS*, pages 759–772, 2015.
18. Lee-Ad Gottlieb and Liam Roditty. An optimal dynamic spanner for doubling metric spaces. In *Algorithms - ESA 2008, 16th Annual European Symposium, Karlsruhe, Germany, September 15-17, 2008. Proceedings*, pages 478–489, 2008. doi:10.1007/978-3-540-87744-8_40.

19 Anupam Gupta, Robert Krauthgamer, and James R. Lee. Bounded geometries, fractals, and low-distortion embeddings. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '03, pages 534–, Washington, DC, USA, 2003. IEEE Computer Society. URL: `http://portal.acm.org/citation.cfm?id=946243.946308`.

20 Sariel Har-Peled and Manor Mendel. Fast construction of nets in low-dimensional metrics and their applications. *SIAM J. Comput.*, 35(5):1148–1184, 2006. `doi:10.1137/S0097539704446281`.

21 William Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. In *Conference in modern analysis and probability (New Haven, Conn., 1982)*, volume 26 of *Contemporary Mathematics*, pages 189–206. American Mathematical Society, 1984.

22 Telikepalli Kavitha. New pairwise spanners. In *32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, March 4-7, 2015, Garching, Germany*, pages 513–526, 2015. `doi:10.4230/LIPIcs.STACS.2015.513`.

23 N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995.

24 J. Matoušek. On the distortion required for embeding finite metric spaces into normed spaces. *Israel Journal of Math*, 93:333–344, 1996.

25 Jiri Matousek. *Lectures on Discrete Geometry*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

26 Manor Mendel and Assaf Naor. Ramsey partitions and proximity data structures. *Journal of the European Mathematical Society*, 9(2):253–275, 2007.

27 Giri Narasimhan and Michiel Smid. *Geometric Spanner Networks*. Cambridge University Press, New York, NY, USA, 2007.

28 Ofer Neiman. Low dimensional embeddings of doubling metrics. *Theory Comput. Syst.*, 58(1):133–152, 2016. `doi:10.1007/s00224-014-9567-3`.

29 Merav Parter. Bypassing erdős' girth conjecture: Hybrid stretch and sourcewise spanners. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part II*, pages 608–619, 2014. `doi:10.1007/978-3-662-43951-7_49`.

30 Neil Robertson and P. D. Seymour. Graph minors: X. obstructions to tree-decomposition. *J. Comb. Theory Ser. B*, 52(2):153–190, 1991. `doi:10.1016/0095-8956(91)90061-N`.

31 Aleksandrs Slivkins. Distance estimation and object location via rings of neighbors. *Distributed Computing*, 19(4):313–333, 2007. `doi:10.1007/s00446-006-0015-8`.

32 Kunal Talwar. Bypassing the embedding: Algorithms for low dimensional metrics. In *Proceedings of the Thirty-sixth Annual ACM Symposium on Theory of Computing*, STOC '04, pages 281–290, New York, NY, USA, 2004. ACM. `doi:10.1145/1007352.1007399`.

Products of Euclidean Metrics and Applications to Proximity Questions among Curves

Ioannis Z. Emiris[1]
Department of Informatics & Telecommunications
National & Kapodistrian University of Athens, Greece, and
ATHENA Research & Innovation Center, Greece
emiris@di.uoa.gr

Ioannis Psarros[2]
Department of Informatics & Telecommunications
National & Kapodistrian University of Athens, Greece
ipsarros@di.uoa.gr

Abstract

The problem of Approximate Nearest Neighbor (ANN) search is fundamental in computer science and has benefited from significant progress in the past couple of decades. However, most work has been devoted to pointsets whereas complex shapes have not been sufficiently treated. Here, we focus on distance functions between discretized curves in Euclidean space: they appear in a wide range of applications, from road segments and molecular backbones to time-series in general dimension. For ℓ_p-products of Euclidean metrics, for any $p \geq 1$, we design simple and efficient data structures for ANN, based on randomized projections, which are of independent interest. They serve to solve proximity problems under a notion of distance between discretized curves, which generalizes both discrete Fréchet and Dynamic Time Warping distances. These are the most popular and practical approaches to comparing such curves. We offer the first data structures and query algorithms for ANN with arbitrarily good approximation factor, at the expense of increasing space usage and preprocessing time over existing methods. Query time complexity is comparable or significantly improved by our algorithms; our approach is especially efficient when the length of the curves is bounded.

2012 ACM Subject Classification Theory of computation → Data structures design and analysis

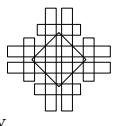
Keywords and phrases Approximate nearest neighbor, polygonal curves, Fréchet distance, dynamic time warping

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.37

1 Introduction

The problem of Approximate Nearest Neighbor (ANN) search is fundamental in computer science: one has to preprocess a dataset so as to answer proximity queries efficiently, for a given query object. ANN has been enjoying a lot of attention and significant progress has been achieved in the past couple of decades. However, most work has been devoted to vector spaces, and complex objects have not been sufficiently treated. Here, we focus on

[1] Emiris is partially supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 734242 (Project LAMBDA).
[2] Psarros is partially supported by a scholarship from the State Scholarships Foundation of Greece, financed by action "Supporting human resources in research through the implementation of doctoral research" in operational program "Human Resources Development, Education and Lifelong Learning", 2014-20, which is co-financed by the European Social Fund and the Greek Government.



distance functions for polygonal curves which lie in the Euclidean space. Polygonal curves are essentially point sequences of varying length and have a wide range of applications ranging from road segments in low dimensions to time-series in arbitrary dimension and protein backbone structures. In general, the problem we aim to solve is as follows.

▶ **Definition 1** (ANN)**.** Input are n polygonal curves $V_1, \ldots, V_n$, where each V_i is a sequence $v_{i1}, \ldots, v_{im_i}$ with each $v_{ij} \in \mathbb{R}^d$, and each $m_i \leq m$ for some pre-specified m. Given distance function $d(\cdot, \cdot)$, $\epsilon > 0$, preprocess $V_1, \ldots, V_n$ into a data structure s.t. for any query polygonal curve Q, the data structure reports V_j for which the following holds

$$\forall i: \ d(Q, V_j) \leq (1+\epsilon) \cdot d(Q, V_i).$$

There are various ways to define dissimilarity or distance between two curves. Two popular dissimilarity measures are the Discrete Fréchet Distance (DFD) and the Dynamic Time Warping (DTW) distance which are both widely studied and applied to classification and retrieval problems for various types of data. DFD satisfies the triangular inequality, unlike DTW.

It is common, in distance functions of curves, to involve the notion of a traversal for two curves. Intuitively, a traversal corresponds to a time plan for traversing the two curves simultaneously, starting from the first point of each curve and finishing at the last point of each curve. With time advancing, the traversal advances in at least one of the two curves. DFD is the minimum over traversals, maximum distance of points while traversing. DTW is the minimum over traversals, sum of distances while traversing.

We denote by ℓ_p^d the normed space $(\mathbb{R}^d, \|\cdot\|_p)$, where for any $x = (x_1, \ldots, x_d) \in \mathbb{R}^d$, $\|x\|_p = (\sum_i |x_i|^p)^{1/p}$. We also use the notation $\tilde{O}(f(d, m, n))$, which hides polylogarithmic factors in $f(d, m, n)$ or polynomial factors in $1/\epsilon$.

1.1 Previous work

The ANN problem has been mainly addressed for datasets consisting of points. Efficient deterministic solutions exist when the dimension is constant, e.g. [5], while for high-dimensional data the state-of-the-art solutions are mainly based on the notion of Locality Sensitive Hashing, e.g. [8, 4], or on random projections, e.g. [1, 2]. Another line of work focuses on subsets of general metrics which satisfy some sort of low intrinsic dimension assumption, e.g. [9]. This is only a small fraction of the body of work on pointsets; however, very little is known about distances between curves.

Let us start with point sequences, which are closely related to curves. For metrics $M_1, \ldots, M_k$, we define the ℓ_p-product of $M_1, \ldots, M_k$ as the metric with domain $M_1 \times \cdots \times M_k$ and distance function

$$d((x_1, \ldots, x_k), (y_1, \ldots, y_k)) = \left(\sum_{i=1}^{k} d_{M_i}^p(x_i, y_i) \right)^{1/p}.$$

If there exists an ANN data structure with approximation factor c for each $M_1, \ldots, M_k$, then one can build a data structure for the ℓ_p-product with approximation factor $O(c \log \log n)$ [3, 10].

Let us now focus on curves: the two existing approaches both solve the approximate near neighbor problem, instead of the optimization ANN. It is known that a data structure for the approximate near neighbor problem can be used as a building block for solving the ANN problem. This procedure has provable guarantees on metrics [8], but it is not clear whether it can be extended to non-metric distances such as the DTW.

The first result for DFD by Indyk [10], defined by any metric $(X, d(\cdot,\cdot))$, achieved approximation factor $O((\log m + \log\log n)^{t-1})$, where m is the maximum length of a curve, and $t > 1$ is a trade-off parameter. The solution is based on an efficient data structure for ℓ_∞-products of arbitrary metrics, and achieves space and preprocessing time in $O(m^2|X|)^{tm^{1/t}} \cdot n^{2t}$, and query time in $(m \log n)^{O(t)}$. Table 1 states these bounds for appropriate $t = 1 + o(1)$, hence a constant approximation factor. It is not clear whether the approach may achieve a $1 + \epsilon$ approximation factor by employing more space.

Quite recently, a new data structure was devised for the DFD of curves defined by the Euclidean metric [7]. The approximation factor is $O(d^{3/2})$. The space required is $O(2^{4md} n \log n + mn)$ and each query costs $O(2^{4md} m \log n)$. They also provide a trade-off between space/query time, and the approximation factor. At the other extreme of this trade-off, they achieve space in $O(n \log n + mn)$, query time in $O(m \log n)$ and approximation factor $O(m)$. Our methods can achieve any user-desired approximation factor at the expense of a reasonable increase in the space and time complexities.

Furthermore, they show in [7] that the result establishing an $O(m)$ approximation extends to DTW, whereas the other extreme of the trade-off has remained open.

Table 1 summarizes space and query time complexities, and approximation factors of the main methods for searching among discrete curves under the two main dissimilarity measures.

1.2 Our contribution

Our first contribution is a simple data structure for the ANN problem in ℓ_p-products of finite subsets of ℓ_2^d, for any p. The key ingredient is a random projection from points in ℓ_2 to points in ℓ_p. Although this has proven a relevant approach for ANN of pointsets, it is quite unusual to employ randomized embeddings from ℓ_2 to ℓ_p, $p > 2$, because such norms are considered "harder" than ℓ_2 in the context of proximity searching. After the random projection, the algorithm "vectorizes" all point sequences. The original problem is then translated to the ANN problem for points in $\ell_p^{d'}$, for $d' \approx d \cdot m$ to be specified later, and can be solved by simple bucketing methods in space $\tilde{O}\left(d' n \cdot (1/\epsilon)^{d'}\right)$ and query time $\tilde{O}(d' \log n)$, which is very efficient when $d \cdot m$ is low.

Then, we present a notion of distance between two polygonal curves, which generalizes both DFD and DTW (for a formal definition see Definition 13). The *ℓ_p-distance* of two curves minimizes, over all traversals, the ℓ_p norm of the vector of all Euclidean distances between paired points. Hence, DFD corresponds to ℓ_∞-distance of polygonal curves, and DTW corresponds to ℓ_1-distance of polygonal curves.

Our main contribution is an ANN structure for the ℓ_p-distance of curves, when $1 \leq p < \infty$. This easily extends to ℓ_∞-distance of curves by solving for the ℓ_p-distance, where p is sufficiently large. Our target are methods with approximation factor $1+\epsilon$. Such approximation factors are obtained for the first time, at the expense of larger space or time complexity. Moreover, a further advantage is that our methods solve ANN directly instead of requiring to reduce it to near neighbor search. While a reduction to the near neighbor problem has provable guarantees on metrics [8], we are not aware of an analogous result for non-metric distances such as the DTW.

Specifically, when $p > 2$, there exists a data structure with space and preprocessing time in

$$\tilde{O}\left(n \cdot \left(\frac{d}{p\epsilon} + 2\right)^{O(dm \cdot \alpha_{p,\epsilon})}\right),$$

Table 1 Summary of previous results compared to this paper's: X denotes the domain set of the input metric. The first method is deterministic while the rest are randomized. All previous results are tuned to optimize the approximation factor. The parameters ρ_u, ρ_q satisfy $(1+\epsilon)\sqrt{\rho_q}+\epsilon\sqrt{\rho_u} \geq \sqrt{1+2\epsilon}$.

	Space	Query	Approx.	Comments
DFD	$O(m^2\lvert X\rvert)^{m^{1-o(1)}} \times O(n^{2-o(1)})$	$(m\log n)^{O(1)}$	$O(1)$	any metric [10]
	$\tilde{O}(2^{4md}n)$	$\tilde{O}(2^{4md}\log n)$	$O\left(d^{3/2}\right)$	ℓ_2^d, [7]
	$\tilde{O}(n)\times\left(\frac{d}{\log m}+2\right)^{O(m\cdot d\log(1/\epsilon))}$	$\tilde{O}(d\cdot 2^{2m}\log n)$	$1+\epsilon$	ℓ_2^d, Thm 16
DTW	$\tilde{O}(mn)$	$O(m\log n)$	$O(m)$	ℓ_2^d, rand.[7]
	$\tilde{O}(n)\times 2^{O(m\cdot d\log(1/\epsilon))}$	$\tilde{O}(d\cdot 2^{2m}\log n)$	$1+\epsilon$	ℓ_2^d, Thm 17
	$\tilde{O}\left(2^{2m}n^{1+\rho_u}\right)$	$\tilde{O}\left(2^{2m}n^{\rho_q}\right)$	$1+\epsilon$	ℓ_2^d, Thm 18

where $\alpha_{p,\epsilon}$ depends only on p,ϵ, and query time in $\tilde{O}(2^{2m}\log n)$. When specialized to DFD and compared to [7], our space and preprocessing time complexity is higher by the exponent $\log(1/\epsilon)$ but our query time is linear instead of being exponential in d.

When $p\in[1,2]$, there exists a data structure with space and preprocessing time in

$$\tilde{O}\left(n\cdot 2^{O(dm\cdot\alpha_{p,\epsilon})}\right),$$

where $\alpha_{p,\epsilon}$ depends only on p,ϵ, and query time in $\tilde{O}\left(2^{2m}\log n\right)$. This leads to the first approach that achieves $1+\epsilon$ approximation for DTW at the expense of space, preprocessing and query time complexities being exponential in m. Hence our method is best suited when the curve size is small.

Our results for DTW and DFD are summarized in Table 1 and juxtaposed to existing approaches in [7, 10].

The rest of the paper is structured as follows. In Section 2, we present a data structure for ANN in ℓ_p-products of ℓ_2, which is of independent interest. In Section 3, we employ this result to address the ℓ_p-distance of curves. We conclude with future work.

2 ℓ_p-products of ℓ_2

In this section, we present a simple data structure for ANN in ℓ_p-products of finite subsets of ℓ_2. Recall that the ℓ_p-product of $X_1,\ldots,X_m$, which are finite subsets of ℓ_2, is a metric space with ground set $X_1\times X_2\times\cdots\times X_m$ and distance function:

$$d((x_1,\ldots,x_m),(y_1,\ldots,y_m)) = \left\|\|x_1-y_1\|_2,\ldots,\|x_m-y_m\|_2\right\|_p = \left(\sum_{i=1}^{m}\|x_i-y_i\|_2^p\right)^{1/p}.$$

For ANN, the algorithm first randomly embeds points from ℓ_2 to ℓ_p. Then, it is easy to translate the original problem to ANN in ℓ_p for large vectors corresponding to point sequences.

2.1 Concentration inequalities

In this subsection, we prove concentration inequalities for central absolute moments of the normal distribution. Most of these results are probably folklore and the reasoning is quite similar to the one followed by proofs of the Johnson-Lindenstrauss lemma, e.g. [11].

The 2-stability property of standard normal variables, along with standard facts about their absolute moments imply the following claim.

▶ **Claim 2.** *Let $v \in \mathbb{R}^d$ and let G be $k \times d$ matrix with i.i.d random variables following $N(0,1)$. Then,*

$$\mathbb{E}\left[\|Gv\|_p^p\right] = c_p \cdot k \cdot \|v\|_2^p,$$

where $c_p = \frac{2^{p/2}\cdot\Gamma\left(\frac{p+1}{2}\right)}{\sqrt{\pi}}$ is a constant depending only on $p > 1$.

Proof. Let $g = (X_1, \ldots, X_d)$ be a vector of random variables which follow $N(0,1)$ and any vector $v \in \mathbb{R}^d$. The 2-stability property of gaussian random variables implies that $\langle g, v\rangle \sim N(0, \|v\|_2^2)$. Recall the following standard fact for central absolute moments of $Z \sim N(0, \sigma^2)$:

$$\mathbb{E}[|Z|^p] = \sigma^p \cdot \frac{2^{p/2}\cdot\Gamma\left(\frac{p+1}{2}\right)}{\sqrt{\pi}}.$$

Hence,

$$\mathbb{E}\left[\|Gv\|_p^p\right] = \mathbb{E}\left[\sum_{i=1}^{k} |\langle g_i, v\rangle|^p\right] = k \cdot \|v\|_2^p \cdot \frac{2^{p/2}\cdot\Gamma\left(\frac{p+1}{2}\right)}{\sqrt{\pi}}.$$ ◀

In the following lemma, we give a simple upper bound on the moment generating function of $|X|^p$, where $X \sim N(0,1)$.

▶ **Lemma 3.** *Let $X \sim N(0,\sigma^2)$, $p \geq 1$, and $t > 0$, then $\mathbb{E}[exp(-t|X|^p)] \leq exp(-t\mathbb{E}[|X|^p] + t^2\mathbb{E}[|X|^{2p}])$.*

Proof. We use the easily verified fact that for any $x \leq 1$, $exp(x) \leq 1 + x + x^2$ and the standard inequality $1 + x \leq \mathrm{e}^x$, for all $x \in \mathbb{R}$.

$$\mathbb{E}\left[\mathrm{e}^{-t|X|^p}\right] \leq 1 - t\cdot\mathbb{E}\left[|X|^p\right] + t^2\cdot\mathbb{E}\left[|X|^{2p}\right] \leq \mathrm{e}^{-t\mathbb{E}[|X|^p]+t^2\mathbb{E}[|X|^{2p}]}.$$ ◀

▶ **Claim 4.** *Let $X \sim N(0,1)$. Then, there exists constant $C > 0$ s.t. for any $p \geq 1$, $\mathbb{E}[|X|^{2p}] \leq C \cdot \mathbb{E}[|X|^p]^2$.*

Proof. In the following, we denote by $f(p) \approx g(p)$ the fact that there exist constants $0 < c < C$ s.t. for any $p > 1$, $f(p) \leq C\cdot g(p)$ and $f(p) \geq c\cdot g(p)$. In addition, $f(p) \gtrsim g(p)$ means that $\exists C > 0$ s.t. $\forall p > 1$, $C \cdot f(p) \geq g(p)$. In other words, $g(p) \approx f(p) \iff g(p) = \Theta(f(p))$ and $f(p) \gtrsim g(p) \iff f(p) = \Omega(g(p))$. In the following we make use of the Stirling approximation and standard facts about moments of normal variables.

$$\mathbb{E}\left[|X|^{2p}\right] = \frac{2^p\cdot\Gamma\left(\frac{2p+1}{2}\right)}{\sqrt{\pi}} \approx (2p-1)!! = \frac{(2p)!}{2^p\cdot p!} \approx \left(\frac{(2p)^{2p}}{\mathrm{e}}\right)\cdot\sqrt{p}\cdot\frac{1}{2^p\cdot\left(\frac{p}{\mathrm{e}}\right)^p} \approx \frac{2^p p^p\sqrt{p}}{\mathrm{e}^p} \approx 2^p\cdot p!$$

$$\mathbb{E}\left[|X|^p\right]^2 \approx ((p-1)!!)^2 \gtrsim \left(2^{p/2+1/2}\cdot\left(\frac{(p/2+1/2)}{\mathrm{e}}\right)^{(p/2+1/2)}\right)^2 \approx 2^p\cdot\frac{p^{p+1}}{\mathrm{e}^{p+1}} \gtrsim 2^p\cdot p!$$ ◀

The following lemma is the main ingredient of our embedding, since it provides us with a lower tail inequality for one projected vector.

▶ **Lemma 5.** *Let G be a $k \times d$ matrix with i.i.d. random variables following $N(0,1)$ and consider vector $v \in \mathbb{R}^d$, s.t. $\|v\|_2 = 1$. For appropriate constant $c' > 1$, for $p \geq 1$ and $\delta \in (0,1)$,*

$$Pr[\|Gv\|_p^p \leq (1-\delta)\cdot\mathbb{E}[\|Gv\|_p^p]] \leq \mathrm{e}^{-c'\cdot k\cdot\delta^2}.$$

Proof. For $X \sim N(0,1)$ and any $t > 0$,

$$\Pr\left[\|Gv\|_p^p \leq (1-\delta)\cdot \mathbb{E}\left[\|Gv\|_p^p\right]\right] \leq \mathbb{E}\left[\mathrm{e}^{-t|X|^p}\right]^k \cdot \mathrm{e}^{(t(1-\delta)k\cdot\mathbb{E}[|X|^p])} \leq$$

$$\leq \mathrm{e}^{k(-t\cdot\mathbb{E}[|X|^p]+t^2\cdot C\cdot\mathbb{E}[|X|^p]^2+t\cdot(1-\delta)\cdot\mathbb{E}[|X|^p])}.$$

The last inequality derives from Claim 4. Now, we set $t = \frac{\delta}{2\sqrt{C}\cdot\mathbb{E}[|X|^p]}$, and we assume wlog $C > 4$. Hence,

$$\Pr[\|Gv\|_p^p \leq (1-\delta)\cdot\mathbb{E}[\|Gv\|_p^p]] \leq \mathrm{e}^{-c'\cdot k\cdot\delta^2},$$

for some constant $c' > 1$. ◀

Finally, we make use of the following one-sided Johnson-Lindenstrauss lemma (see e.g. [11]).

▶ **Theorem 6.** *Let G be a $k \times d$ matrix with i.i.d. random variables following $N(0,1)$ and consider vector $v \in \mathbb{R}^d$. Then, for constant $C > 0$,*

$$\Pr\left[\|Gv\|_2 \geq (1+\epsilon)\|v\|_2\sqrt{k}\right] \leq \mathrm{e}^{-C\cdot k\cdot\epsilon^2}.$$

Standard properties of ℓ_p norms imply a loose upper tail inequality.

▶ **Corollary 7.** *Let G be a $k \times d$ matrix with i.i.d. random variables following $N(0,1)$ and consider vector $v \in \mathbb{R}^d$. Let $p \geq 2$. Then, for constant $C > 0$,*

$$\Pr\left[\|Gv\|_p \geq (1+\epsilon)\|v\|_2\sqrt{k}\right] \leq \mathrm{e}^{-C\cdot k\cdot\epsilon^2}.$$

Proof. Since $p \geq 2$, we have that $\forall x \in \mathbb{R}^d\ \|x\|_p \leq \|x\|_2$. Hence, by Theorem 6,

$$\Pr[\|Gv\|_p \geq (1+\epsilon)\|v\|_2\sqrt{k}] \leq \Pr[\|Gv\|_2 \geq (1+\epsilon)\|v\|_2\sqrt{k}] \leq \mathrm{e}^{-C\cdot k\cdot\epsilon^2}.$$ ◀

However, an improved upper tail inequality can be derived when $p \in [1,2]$.

▶ **Lemma 8.** *Let G be a $k \times d$ matrix with i.i.d. random variables following $N(0,1)$ and consider vector $v \in \mathbb{R}^d$. Let $p \in [1,2]$. Then, for constant $C > 0$,*

$$\Pr\left[\|Gv\|_p \geq (3\cdot c_p\cdot k)^{1/p}\|v\|_2\right] \leq \mathrm{e}^{-C\cdot k}.$$

Proof. Let $X \sim N(0,1)$.

$$\mathbb{E}\left[\mathrm{e}^{|X|^p/3}\right] = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{+\infty}\mathrm{e}^{|x|^p/3-x^2/2}\mathrm{d}x \leq \frac{\sqrt{2}}{\sqrt{\pi}}\int_0^{+\infty}\mathrm{e}^{x^2/3-x^2/2}\mathrm{d}x = \sqrt{3}.$$

Now, assume wlog $\|v\|_2 = 1$,

$$\Pr\left[\|Gv\|_p^p \geq 3\cdot\mathbb{E}\left[\|Gv\|_p^p\right]\right] \leq \mathbb{E}\left[\mathrm{e}^{|X|^p/3}\right]^k \cdot \mathrm{e}^{-k\cdot\mathbb{E}[|X|^p]} \leq \mathrm{e}^{-k(c_p-2/3)} \leq \mathrm{e}^{-k/10},$$

where $c_p = \frac{2^{p/2}\cdot\Gamma\left(\frac{p+1}{2}\right)}{\sqrt{\pi}}$. ◀

2.2 Embedding ℓ_2 into ℓ_p

In this subsection, we present our main results concerning ANN for ℓ_p-products of ℓ_2. First, we show that a simple random projection maps points from ℓ_2^d to ℓ_p^k, where $k = \tilde{O}(d)$, without arbitrarily contracting norms. The probability of failure decays exponentially with k. For our purposes, there is no need for an almost isometry between norms. Hence, our efforts focus on proving lower tail inequalities which imply that, with good probability, no far neighbor corresponds to an approximate nearest neighbor in the projected space.

We now prove bounds concerning the contraction of distances of the embedded points. Our proof builds upon the inequalities developed in Subsection 2.1.

▶ **Theorem 9.** *Let G be a $k \times d$ matrix with i.i.d. random variables following $N(0,1)$. Then,*

- *if $2 < p < \infty$ then,*

$$\Pr\left[\exists v \in \mathbb{R}^d : \ \|Gv\|_p \leq \frac{(c_p \cdot k)^{1/p}}{1+\epsilon} \cdot \|v\|_2\right] \leq O\left(\frac{k^{\frac{1}{2}-\frac{1}{p}}}{p\epsilon} + 2\right)^d \cdot \mathrm{e}^{-c' \cdot k \cdot (p\epsilon/(2+p\epsilon))^2},$$

- *if $p \in [1,2]$ then,*

$$\Pr\left[\exists v \in \mathbb{R}^d : \ \|Gv\|_p \leq \frac{(c_p \cdot k)^{1/p}}{1+\epsilon} \cdot \|v\|_2\right] \leq O\left(\frac{1}{\epsilon}\right)^d \cdot \mathrm{e}^{-c' \cdot k \cdot (p\epsilon/(2+p\epsilon))^2},$$

where $c' > 1$ is a constant, $\epsilon \in (0, 1/2)$.

Proof. By Lemma 5:

$$\Pr\left[\|Gv\|_p^p \leq \frac{c_p \cdot k}{(1+\epsilon)^p} \cdot \|v\|_2^p\right] \leq \Pr\left[\|Gv\|_p^p \leq \frac{c_p \cdot k}{1+p\epsilon/2} \cdot \|v\|_2^p\right] \leq \mathrm{e}^{-c' \cdot k \cdot (p\epsilon/(2+p\epsilon))^2}.$$

In order to bound the probability of contraction among all distances, we argue that it suffices to use the strong bound on distance contraction, which is derived in Lemma 5, and the weak bound on distance expansion from Corollary 7 or Lemma 8, for a δ-dense set $N \subset \mathbb{S}^{d-1}$ for δ to be specified later. First, a simple volumetric argument [8] shows that there exists $N \subset \mathbb{S}^{d-1}$ s.t. $\forall x \in \mathbb{S}^{d-1}\ \exists y \in N\ \|x-y\|_2 \leq \delta$, and $|N| = O\left(1/\delta\right)^d$.

We first consider the case $p > 2$. From now on, we assume that for any $u \in N$, $\|Gu\|_p \geq (c_p \cdot k)^{1/p}/(1+\epsilon)$ and $\|Gu\|_p \leq 2\sqrt{k}$ which is achieved with probability

$$\geq 1 - O\left(\frac{1}{\delta}\right)^d \cdot \mathrm{e}^{-c' \cdot k \cdot (p\epsilon/(2+p\epsilon))^2}.$$

Now let x be an arbitrary vector in $\mathbb{R}^d$ s.t. $\|x\|_2 = 1$. Then, there exists $u \in N$ s.t. $\|x-u\|_2 \leq \delta$. Also, by the triangular inequality we obtain the following,

$$\|Gx\|_p \leq \|Gu\|_p + \|G(x-u)\|_p = \|Gu\|_p + \|x-u\|_2 \left\|G\frac{(x-u)}{\|x-u\|_2}\right\|_p \implies$$

$$\implies \|Gx\|_p \leq \|Gu\|_p + \delta \left\|G\frac{(x-u)}{\|x-u\|_2}\right\|_p. \tag{1}$$

Let $M = \max_{x \in \mathbb{S}^{d-1}} \|Gx\|_p$. The existence of M is implied by the fact that $\mathbb{S}^{d-1}$ is compact and $x \mapsto \|x\|_p$, $x \mapsto Gx$ are continuous functions. Then, by plugging M into (1),

$$M \leq \|Gu\|_p + \delta M \implies M \leq \frac{\|Gu\|_p}{1-\delta} \leq \frac{2\sqrt{k}}{1-\delta},$$

where the last inequality is implied by Corollary 7. Again, by the triangular inequality,

$$\|Gx\|_p \geq \|G_u\|_p - \|G(x-u)\|_p \geq \frac{(c_p \cdot k)^{1/p}}{1+\epsilon} - \frac{2\delta\sqrt{k}}{1-\delta} \geq \frac{1-\epsilon/2}{1+\epsilon} \cdot (c_p \cdot k)^{1/p},$$

for $\delta \leq \frac{\epsilon \cdot (c_p \cdot k)^{1/p}}{2\sqrt{k}+\epsilon \cdot (c_p \cdot k)^{1/p}}$.

Notice now that

$$\frac{1}{\delta} = O\left(\frac{k^{1/2-1/p}}{p\epsilon}\right) + 1.$$

In the case $p \in [1,2]$, we are able to use a better bound on the distance expansion; namely Lemma 8. We now assume that for any $u \in N$, $\|Gu\|_p \geq (c_p \cdot k)^{1/p}/(1+\epsilon)$ and $\|Gu\|_p \leq (3 \cdot c_p \cdot k)^{1/p}$ which is achieved with probability

$$\geq 1 - O\left(\frac{1}{\delta}\right)^d \cdot \mathrm{e}^{-c' \cdot k \cdot (p\epsilon/(2+p\epsilon))^2}.$$

Once again, we use inequality (1) to obtain:

$$M \leq \frac{\|Gu\|_p}{1-\delta} \leq \frac{(3 \cdot c_p \cdot k)^{1/p}}{1-\delta} \implies$$

$$\implies \|Gx\|_p \geq \|Gu\|_p - \|Gx - Gu\|p \geq (c_p \cdot k)^{1/p}\left(\frac{1}{1+\epsilon} - \frac{3^{1/p} \cdot \delta}{1-\delta}\right) \implies$$

$$\implies \|Gx\|_p \geq (c_p \cdot k)^{1/p} \cdot \frac{1-\epsilon/2}{1+\epsilon},$$

for $\delta \leq \epsilon/(6(1+\epsilon)+\epsilon) = \Omega(\epsilon)$. ◀

Theorem 9 implies that the ANN problem for ℓ_p products of ℓ_2 translates to the ANN problem for ℓ_p products of ℓ_p. The latter easily translates to the ANN problem in $\ell_p^{d'}$. One can then solve the approximate near neighbor problem in $\ell_p^{d'}$, by approximating $\ell_p^{d'}$ balls of radius 1 with a regular grid with side length $\epsilon/(d')^{1/p}$. Each approximate ball is essentially a set of $O(1/\epsilon)^{d'}$ cells [8]. Building not-so-many approximate near neighbor data structures for various radii leads to an efficient solution for the ANN problem [8].

▶ **Theorem 10.** *There exists a data structure which solves the ANN problem for point sequences in ℓ_p-products of ℓ_2, and satisfies the following bounds on performance:*

- *If $p \in [1,2]$, then space usage and preprocessing time is in*

$$\tilde{O}(dmn) \times \left(\frac{1}{\epsilon}\right)^{O(m \cdot d \cdot \alpha_{p,\epsilon})},$$

  query time is in $\tilde{O}(dm \log n)$, and $\alpha_{p,\epsilon} = \log(1/\epsilon) \cdot (2+p\epsilon)^2 \cdot (p\epsilon)^{-2}$.
- *If $2 < p < \infty$, then space usage and preprocessing time is in*

$$\tilde{O}(dmn) \times \left(\frac{d}{p\epsilon} + 2\right)^{O(m \cdot d \cdot \alpha_{p,\epsilon})},$$

  query time is in $\tilde{O}(dm \log n)$, and $\alpha_{p,\epsilon} = \log(1/\epsilon) \cdot (2+p\epsilon)^2 \cdot (p\epsilon)^{-2}$.

We assume $\epsilon \in (0, 1/2]$. The probability of success is $\Omega(\epsilon)$ and can be amplified to $1-\delta$, by building $\Omega(\log(1/\delta)/\epsilon)$ independent copies of the data-structure.

Proof. Let $\delta_{p,\epsilon} = p\epsilon/(2+p\epsilon)$. We first consider the case $p > 2$. We employ Theorem 9 and we map point sequences to point sequences in ℓ_p^k, for

$$k = \Theta\left(\frac{d \log \frac{d}{p\epsilon}}{\delta_{p,\epsilon}^2}\right).$$

Hence, Theorem 9 implies that,

$$\Pr\left[\exists v \in \mathbb{R}^d : \ \|Gv\|_p \leq \frac{(c_p \cdot k)^{1/p}}{1+\epsilon} \cdot \|v\|_2\right] \leq \epsilon/10.$$

Then, by concatenating vectors, we map point sequences to points in ℓ_p^{km}.

Now, fix query point sequence $Q = q_1, \ldots, q_m \in \left(\mathbb{R}^d\right)^m$ and its nearest neighbor $U_* = u_1, \ldots, u_m \in \left(\mathbb{R}^d\right)^m$. By a union bound, the probability of failure for the embedding is at most

$$\epsilon/2 + \Pr\left[\sum_{i=1}^m \|Gu_i - Gq_i\|_p^p \leq (1+\epsilon)^p \cdot c_p \cdot k \sum_{i=1}^m \|u_i - q_i\|_2^p\right],$$

since we already know that

$$\Pr\left[\exists v \in \mathbb{R}^d : \ \|Gv\|_p \leq \frac{(c_p \cdot k)^{1/p}}{1+\epsilon} \cdot \|v\|_2\right] \leq \epsilon/2.$$

Hence, we now bound the second probability. Notice that

$$\mathbb{E}\left[\sum_{i=1}^m \|Gu_i - Gq_i\|_p^p\right] = \sum_{i=1}^m \mathbb{E}\left[\|G(u_i - q_i)\|_p^p\right] = c_p \cdot k \sum_{i=1}^m \|u_i - q_i\|_2^p.$$

By Markov's inequality, we obtain,

$$Pr\left[\sum_{i=1}^m \|Gu_i - Gq_i\|_p^p \leq (1+\epsilon)^p \cdot c_p \cdot k \sum_{i=1}^m \|u_i - q_i\|_2^p\right] \leq (1+\epsilon)^{-p}.$$

Hence, the total probability of failure is $\frac{1+\epsilon/10}{(1+\epsilon)^p}$. In the projected space, we build AVDs[8]. The total space usage, and the preprocessing time is

$$\tilde{O}(dmn) \times O(1/\epsilon)^{km} = \tilde{O}(dmn) \times \left(\frac{d}{p\epsilon} + 2\right)^{O(m \cdot d \cdot \log(1/\epsilon)/\delta_{p,\epsilon}^2)}.$$

The query time is $\tilde{O}(dm \log n)$. The probability of success can be amplified by repetition. By building $\Theta\left(\frac{\log(1/\delta)}{\epsilon}\right)$ data structures as above, the probability of failure becomes δ.

The same reasoning is valid in the case $p \in [1,2]$, but it suffices to set

$$k = \Theta\left(\frac{d \log \frac{1}{\epsilon}}{\delta_{p,\epsilon}^2}\right). \qquad \blacktriangleleft$$

When $p \in [1,2]$, we can also utilize "high-dimensional" solutions for ℓ_p and obtain data structures with complexities polynomial in $d \cdot m$. Combining Theorem 9 with the data structure of [4], we obtain the following result.

▶ **Theorem 11.** *There exists a data structure which solves the ANN problem for point sequences in ℓ_p-products of ℓ_2, $p \in [1,2]$, and satisfies the following bounds on performance: space usage and preprocessing time is in $\tilde{O}(n^{1+\rho_u})$, and the query time is in $\tilde{O}(n^{\rho_q})$, where ρ_q, ρ_u satisfy:*

$$(1+\epsilon)^p\sqrt{\rho_q} + ((1+\epsilon)^p - 1)\sqrt{\rho_u} \geq \sqrt{2(1+\epsilon)^p - 1}$$

We assume $\epsilon \in (0, 1/2]$. The probability of success is $\Omega(\epsilon)$ and can be amplified to $1-\delta$, by building $\Omega(\log(1/\delta)/\epsilon)$ independent copies of the data-structure.

Proof. We proceed as in the proof of Theorem 10. We employ Theorem 9 and by Markov's inequality, we obtain,

$$Pr\left[\sum_{i=1}^{m} \|Gv_i - Gu_i\|_p^p \leq (1+\epsilon)^p \cdot c_p \cdot k \sum_{i=1}^{m} \|v_i - u_i\|_2^p\right] \leq (1+\epsilon)^{-p}.$$

Then, by concatenating vectors, we map point sequences to points in ℓ_p^{km}, where $k = \tilde{O}(d)$. For the mapped points in ℓ_p^{km}, we build the LSH-based data structure from [4] which succeeds with high probability $1 - o(1)$. By independence, both the random projection and the LSH-based structure succeed with probability $\Omega(\epsilon) \times (1 - o(1)) = \Omega(\epsilon)$. ◀

3 Polygonal curves

In this section, we show that one can solve the ANN problem for a certain class of distance functions defined on polygonal curves. Since this class is related to ℓ_p-products of ℓ_2, we invoke results of Section 2, and we show an efficient data structure for the case of short curves, i.e. when m is relatively small compared to the other complexity parameters.

First, we need to introduce a formal definition of the traversal of two curves.

▶ **Definition 12.** Given polygonal curves $V = v_1, \ldots, v_{m_1}$, $U = u_1, \ldots, u_{m_2}$, a traversal $T = (i_1, j_1), \ldots, (i_t, j_t)$ is a sequence of pairs of indices referring to a pairing of vertices from the two curves such that:

1. $i_1, j_1 = 1$, $i_t = m_1$, $j_t = m_2$.
2. $\forall (i_k, j_k) \in T : i_{k+1} - i_k \in \{0, 1\}$ and $j_{k+1} - j_k \in \{0, 1\}$.
3. $\forall (i_k, j_k) \in T : (i_{k+1} - i_k) + (j_{k+1} - j_k) \geq 1$.

Now, we define a class of distance functions for polygonal curves. In this definition, it is implied that we use the Euclidean distance to measure distance between any two points. However, the definition could be easily generalized to arbitrary metrics.

▶ **Definition 13** (ℓ_p-distance of polygonal curves)**.** Given polygonal curves $V = v_1, \ldots, v_{m_1}$, $U = u_1, \ldots, u_{m_2}$, we define the ℓ_p-distance between V and U as the following function:

$$d_p(V, U) = \min_{T \in \mathcal{T}} \left(\sum_{(i_k, j_k) \in T} \|v_{i_k} - u_{j_k}\|_2^p \right)^{1/p},$$

where $\mathcal{T}$ denotes the set of all possible traversals for V and U.

The above class of distances for curves includes some widely known distance functions. For instance, $d_\infty(V, U)$ coincides with the DFD of V and U (defined for the Euclidean distance). Moreover $d_1(V, U)$ coincides with DTW for curves V, U.

▶ **Theorem 14.** *Suppose that there exists a randomized data structure for the ANN problem in ℓ_p products of ℓ_2, with space in $S(n)$, preprocessing time $T(n)$ and query time $Q(n)$, with probability of failure less than $2^{-2m-1}m^{-1}$. Then, there exists a data structure for the ANN problem for the ℓ_p-distance of polygonal curves, $1 \leq p < \infty$, with space in $m \cdot 2^{2m} \cdot S(n)$, preprocessing time $m \cdot 2^{2m} \cdot T(n)$ and query time $m \cdot 2^{2m} \cdot Q(n)$, where m denotes the maximum length of a polygonal curve, and the probability of failure is less than $1/2$.*

Proof. We denote by X the input dataset. Given polygonal curves $V = v_1, \ldots, v_{m_1}$, $Q = q_1, \ldots, q_{m_2}$, and traversal T, one can define $V_T = v_1, \ldots, v_l$, $Q_T = q_1, \ldots, q_l$, sequences of l points (allowing consecutive duplicates) s.t. $\forall k$, $v_{i_k} = V_T[k]$ and $q_{j_k} = Q_T[k]$, if and only if $(i_k, j_k) \in T$.

One traversal of V, Q is uniquely defined by its length $l \in \{\max(m_1, m_2), \ldots, m_1 + m_2\}$, the set of indices $A = \{k \in \{1, \ldots, l\} \mid i_{k+1} - i_k = 0 \text{ and } j_{k+1} - j_k = 1\}$ for which only Q is progressing and the set of indices $B = \{k \in \{1, \ldots, l\} \mid i_{k+1} - i_k = 1 \text{ and } j_{k+1} - j_k = 1\}$ for which both Q and V are progressing. We can now define $V_{l,A,B}$, $Q_{l,A,B}$ to be the corresponding sequences of l points. In other words if l, A, B corresponds to traversal T, $V_{l,A,B} = V_T$, $Q_{l,A,B} = Q_T$. Observe that it is possible that curve V is not compatible with some triple l, A, B.

We build one ANN data structure, for ℓ_p products of ℓ_2, for each possible l, A, B. Each data structure contains at most $|X|$ point sequences which correspond to curves that are compatible to l, A, B. We denote by $m = \max(m_1, m_2)$. The total number of data structures is upper bounded by

$$\sum_{l=m}^{2m} \sum_{t=0}^{m} \binom{l}{t} \cdot \binom{l-t}{m-t} \leq \sum_{l=m}^{2m} \sum_{t=0}^{m} \binom{l}{t} \cdot \binom{l}{m-t} = \sum_{l=m}^{2m} \binom{2l}{m} \leq m \cdot \binom{2m}{m} \leq m \cdot 2^{2m}.$$

For any query curve Q, we create all possible combinations of l, A, B and we perform one query per ANN data structure. We report the best answer. The probability that the building of one of the $\leq m \cdot 2^{2m}$ data structures is not successful is less than $1/2$ due to a union bound. ◀

We now investigate applications of the above results, to the ANN problem for some popular distance functions for curves.

Discrete Fréchet Distance

DFD is naturally included in the distance class of Definition 13 for $p = \infty$. However, Theorem 14 is valid only when p is bounded. To overcome this issue, p is set to a suitable large value.

▶ **Claim 15.** *Let $V = v_1, \ldots, v_{m_1} \in \mathbb{R}^d$ and $U = u_1, \ldots, u_{m_2} \in \mathbb{R}^d$ be two polygonal curves. Then for any traversal T of V and U:*

$$(1+\epsilon)^{-1} \cdot \left(\sum_{(i_k,j_k)\in T} \|v_{i_k} - u_{j_k}\|^p \right)^{1/p} \leq \max_{(i_k,j_k)\in T} \|v_{i_k} - u_{j_k}\| \leq \left(\sum_{(i_k,j_k)\in T} \|v_{i_k} - u_{j_k}\|^p \right)^{1/p},$$

for $p \geq \log(|T|) / \log(1+\epsilon)$.

Proof. For any $x \in \mathbb{R}^{|T|}$, it is known that $\|x\|_\infty \leq \|x\|_p \leq (|T|)^{1/p} \|x\|_\infty$. ◀

▶ **Theorem 16.** *There exists a data structure for the ANN problem for the DFD of curves, with space and preprocessing time*

$$\tilde{O}(dm^2n) \times \left(\frac{d}{\log m} + 2\right)^{O(m \cdot d \cdot \log(1/\epsilon))},$$

and query time $\tilde{O}(d \cdot 2^{2m} \log n)$, where m denotes the maximum length of a polygonal curve, and $\epsilon \in (0, 1/2]$. The data structure succeeds with probability $1/2$, which can be amplified by repetition.

Proof. We combine Theorem 14 with Theorem 10 for $p \geq \log m / \log(1+\epsilon) \geq \epsilon^{-1} \log m$. Notice that in order to plug the data structure of Theorem 10 into Theorem 14 we need to amplify the probability of success to $1 - 2^{-2m-1} \cdot m^{-1}$. Hence, the data structure for the ANN problem for ℓ_p-products of ℓ_p needs space and preprocessing time

$$\tilde{O}(dm^2n) \times \left(\frac{d}{p\epsilon} + 2\right)^{O(m \cdot d \cdot \alpha_{p,\epsilon})},$$

and each query time costs $O(dm^2)$, where $\alpha_{p,\epsilon} = \log(1/\epsilon) \cdot (2+p\epsilon)^2 \cdot (p\epsilon)^{-2}$. Now, substituting p and invoking Theorem 14 completes our proof. ◀

Dynamic Time Warping

DTW corresponds to the ℓ_1-distance of polygonal curves as defined in Definition 13. Now, we combine Theorem 14 with each of the Theorems 10 and 11.

▶ **Theorem 17.** *There exists a data structure for the ANN problem for DTW of curves, with space and preprocessing time*

$$\tilde{O}(dm^2n) \times \left(\frac{1}{\epsilon}\right)^{O(m \cdot d \cdot \epsilon^{-2})},$$

and query time $\tilde{O}(d \cdot 2^{2m} \log n)$, where m denotes the maximum length of a polygonal curve, and $\epsilon \in (0, 1/2]$. The data structure succeeds with probability $1/2$, which can be amplified by repetition.

Proof. We first amplify the probability of success for the data structure of Theorem 10 to $1 - 2^{-2m-1} \cdot m^{-1}$. Hence, the data structure for the ANN problem for ℓ_1-products of ℓ_1 needs space and preprocessing time

$$\tilde{O}(dm^2n) \times 2^{O(m \cdot d \cdot \alpha_{p,\epsilon})},$$

and each query time costs $O(dm^2)$, where $\alpha_{p,\epsilon} = \log(1/\epsilon) \cdot (2+\epsilon)^2 \cdot (\epsilon)^{-2}$. We plug this data structure into Theorem 14. ◀

▶ **Theorem 18.** *There exists a data structure for the ANN problem for DTW of curves, with space and preprocessing time $\tilde{O}(2^{2m}n^{1+\rho_u})$, and the query time is in $\tilde{O}(2^{2m}n^{\rho_q})$, where ρ_q, ρ_u satisfy:*

$$(1+\epsilon)\sqrt{\rho_q} + \epsilon\sqrt{\rho_u} \geq \sqrt{1+2\epsilon}.$$

We assume $\epsilon \in (0, 1/2]$. The data structure succeeds with probability $1/2$, which can be amplified by repetition.

Proof. First amplify the probability of success for the data structure of Theorem 11 to $1 - 2^{-2m-1} \cdot m^{-1}$, by building independently $\tilde{O}(m)$ such data structures. We plug the resulting data structure into Theorem 14. ◀

4 Conclusion

Thanks to the simplicity of the approach, it should be easy to implement it and should have practical interest. We plan to apply it to real scenarios with data from road segments or time series.

The key ingredient of our approach is a randomized embedding from ℓ_2 to ℓ_p which is the first step to the ANN solution for ℓ_p-products of ℓ_2. The embedding is essentially a gaussian projection and it exploits the 2-stability property of normal variables, along with standard properties of their tails. We expect that a similar result can be achieved for ℓ_p-products of ℓ_q, where $q \in [1, 2)$. One related result for ANN [6], provides with dimension reduction for ℓ_q, $q \in [1, 2)$.

References

1 E. Anagnostopoulos, I. Z. Emiris, and I. Psarros. Low-quality dimension reduction and high-dimensional approximate nearest neighbor. In *Proc. 31st Intern. Symp. on Computational Geometry (SoCG)*, pages 436–450, 2015. doi:10.4230/LIPIcs.SOCG.2015.436.

2 E. Anagnostopoulos, I. Z. Emiris, and I. Psarros. Randomized embeddings with slack, and high-dimensional approximate nearest neighbor. In *ACM Transactions on Algorithm*, 2018, To appear.

3 A. Andoni. *NN search: the old, the new, and the impossible*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2009. URL: http://hdl.handle.net/1721.1/55090.

4 A. Andoni, T. Laarhoven, I. Razenshteyn, and E. Waingarten. Optimal hashing-based time-space trade-offs for approximate near neighbors. In *Proc. Symp. Discrete Algorithms (SODA)*, 2017. Also as arxiv.org/abs/1608.03580.

5 S. Arya, T. Malamatos, and D. M. Mount. Space-time tradeoffs for approximate nearest neighbor searching. *J. ACM*, 57(1):1:1–1:54, 2009. doi:10.1145/1613676.1613677.

6 Y. Bartal and L. -A. Gottlieb. Dimension reduction techniques for p (1<p<2), with applications. In *32nd International Symposium on Computational Geometry, SoCG 2016, June 14-18, 2016, Boston, MA, USA*, pages 16:1–16:15, 2016. doi:10.4230/LIPIcs.SoCG.2016.16.

7 A. Driemel and F. Silvestri. Locality-sensitive hashing of curves. In *Proc. 33rd Intern. Symposium on Computational Geometry*, pages 37:1–37:16, 2017. doi:10.4230/LIPIcs.SoCG.2017.37.

8 S. Har-Peled, P. Indyk, and R. Motwani. Approximate nearest neighbor: Towards removing the curse of dimensionality. *Theory of Computing*, 8(1):321–350, 2012. doi:10.4086/toc.2012.v008a014.

9 S. Har-Peled and M. Mendel. Fast construction of nets in low dimensional metrics, and their applications. In *Proc. 21st Annual Symp. Computational Geometry*, SCG'05, pages 150–158, 2005. doi:10.1145/1064092.1064117.

10 P. Indyk. Approximate nearest neighbor algorithms for frechet distance via product metrics. In *Proc. 18th Annual Symp. on Computational Geometry*, SCG '02, pages 102–106, New York, NY, USA, 2002. ACM. doi:10.1145/513400.513414.

11 J. Matoušek. On variants of the Johnson-Lindenstrauss lemma. *Random Struct. Algorithms*, 33(2):142–156, 2008. doi:10.1002/rsa.v33:2.

Rainbow Cycles in Flip Graphs

Stefan Felsner
Institut für Mathematik, TU Berlin, Germany
felsner@math.tu-berlin.de

Linda Kleist
Institut für Mathematik, TU Berlin, Germany
kleist@math.tu-berlin.de

Torsten Mütze
Institut für Mathematik, TU Berlin, Germany
muetze@math.tu-berlin.de

Leon Sering
Institut für Mathematik, TU Berlin, Germany
sering@math.tu-berlin.de

Abstract

The flip graph of triangulations has as vertices all triangulations of a convex n-gon, and an edge between any two triangulations that differ in exactly one edge. An r-rainbow cycle in this graph is a cycle in which every inner edge of the triangulation appears exactly r times. This notion of a rainbow cycle extends in a natural way to other flip graphs. In this paper we investigate the existence of r-rainbow cycles for three different flip graphs on classes of geometric objects: the aforementioned flip graph of triangulations of a convex n-gon, the flip graph of plane spanning trees on an arbitrary set of n points, and the flip graph of non-crossing perfect matchings on a set of n points in convex position. In addition, we consider two flip graphs on classes of non-geometric objects: the flip graph of permutations of $\{1, 2, \ldots, n\}$ and the flip graph of k-element subsets of $\{1, 2, \ldots, n\}$. In each of the five settings, we prove the existence and non-existence of rainbow cycles for different values of r, n and k.

2012 ACM Subject Classification Mathematics of computing → Combinatorics, Mathematics of computing → Permutations and combinations, Mathematics of computing → Graph theory, Theory of computation → Randomness, geometry and discrete structures

Keywords and phrases flip graph, cycle, rainbow, Gray code, triangulation, spanning tree, matching, permutation, subset, combination

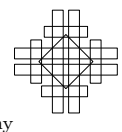
Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.38

Related Version A full version of this paper is available at `https://arxiv.org/abs/1712.07421`

Acknowledgements We thank Manfred Scheucher for his quick assistance in running computer experiments that helped us to find rainbow cycles in small flip graphs.

1 Introduction

Flip graphs are fundamental structures associated with families of geometric objects such as triangulations, plane spanning trees, non-crossing matchings, partitions or dissections. A classical example is the flip graph of triangulations. The vertices of this graph G_n^{T} are the triangulations of a convex n-gon, and two triangulations are adjacent whenever they differ by exactly one edge. In other words, moving along an edge of G_n^{T} corresponds to flipping the diagonal of a convex quadrilateral formed by two triangles. Figure 1 shows the graph G_6^{T}.



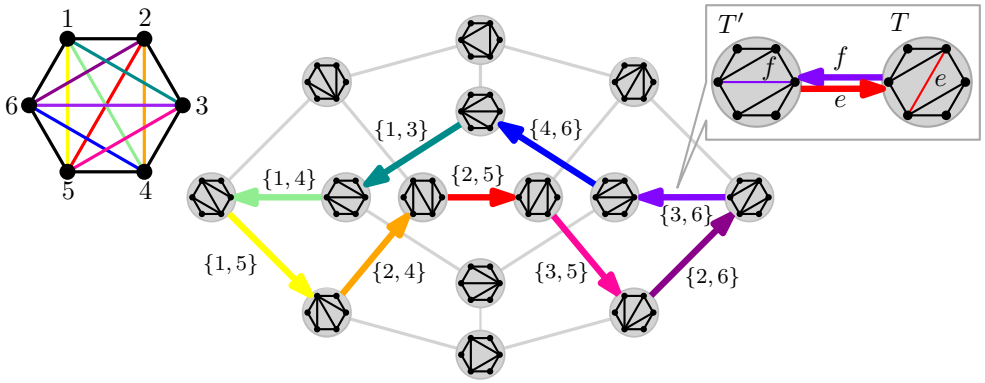
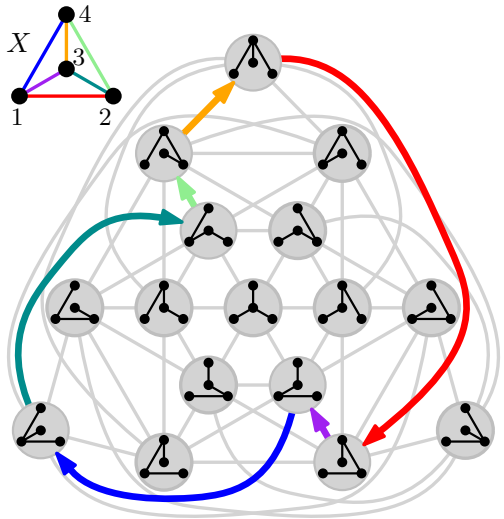


Figure 1 The flip graph of triangulations G_6^{T} with a highlighted rainbow cycle.

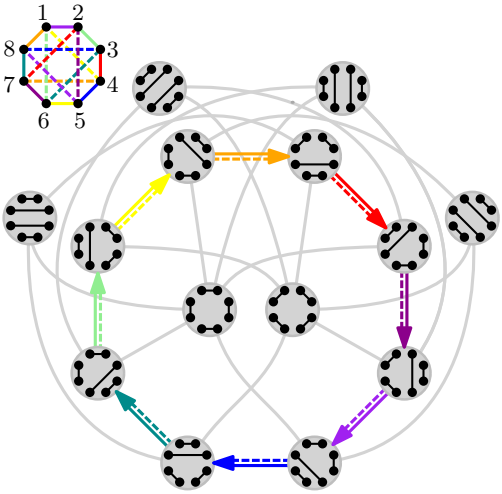
A question that has received considerable attention is to determine the diameter of G_n^{T}, i.e., the number of flips that is necessary and sufficient to transform any triangulation into any other, see the survey [7]. In a landmark paper [33], Sleator, Tarjan and Thurston proved that the diameter of G_n^{T} is $2n - 10$ for sufficiently large n. Recently, Pournin [28] gave a combinatorial proof that the diameter is $2n - 10$ for all $n > 12$. A challenging algorithmic problem in this direction is to efficiently compute a minimal sequence of flips that transforms two given triangulations into each other, see [24, 29]. These questions involving the diameter of the flip graph become even harder when the n points are not in convex, but in general position, see e.g. [11, 15, 19]. Moreover, apart from the diameter, many other properties of the flip graph G_n^{T} have been investigated, e.g., its realizability as a convex polytope [9], its automorphism group [23], the vertex-connectivity [18], and the chromatic number [12].

Another property of major interest is the existence of a Hamilton cycle in G_n^{T}. This was first established by Lucas [25] and a very nice and concise proof was given by Hurtado and Noy [18]. The reason for the interest in Hamilton cycles is that a Hamilton cycle in G_n^{T} corresponds to a so-called *Gray code*, i.e., an algorithm that allows to generate each triangulation exactly once, by performing only a single flip operation when moving to the next triangulation. In general, the task of a Gray code algorithm is to generate all objects in a particular combinatorial class, each object exactly once, by applying only a small transformation in each step, such as a flip in a triangulation. Combinatorial classes of interest include geometric configurations such as triangulations, plane spanning trees or non-crossing perfect matchings, but also classes without geometric information such as permutations, combinations, bitstrings etc. This fundamental topic is covered in depth in the most recent volume of Knuth's seminal series *The Art of Computer Programming* [21], and in the classical books by Nijenhuis and Wilf [27, 36]. Here are some important Gray code results in the geometric realm: Hernando, Hurtado and Noy [16] proved the existence of a Hamilton cycle in the flip graph of non-crossing perfect matchings on a set of $2m$ points in convex position for every even $m \geq 4$. Aichholzer et al. [1] described Hamilton cycles in the flip graphs of plane graphs on a general point set, for plane and connected graphs and for plane spanning trees on a general point set. Huemer et al. [17] constructed Hamilton cycles in the flip graphs of non-crossing partitions of a point set in convex position, and for the dissections of a convex polygon by a fixed number of non-crossing diagonals.

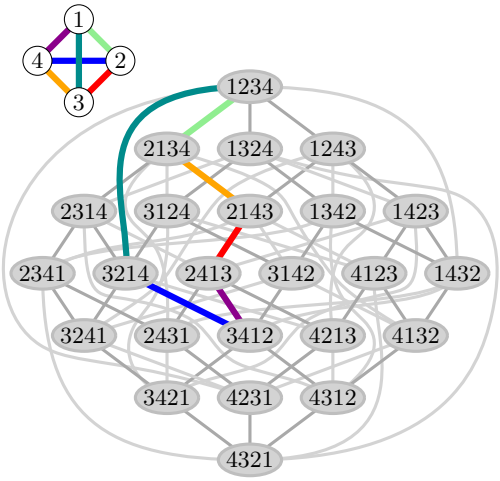
As mentioned before, a Hamilton cycle in a flip graph corresponds to a cyclic listing of all objects in some combinatorial class, such that each object is encountered exactly once, by performing a single flip in each step. In this work we consider the *dual* problem: we are



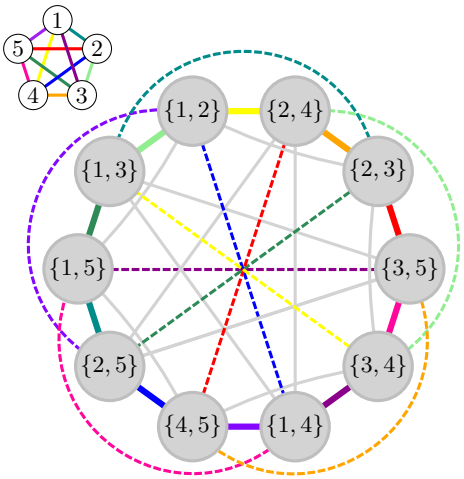
(a) Flip graph of plane spanning trees G_X^{S}.



(b) Flip graph of non-crossing perfect matchings G_4^{M}.



(c) Flip graph of permutations G_4^{P}.



(d) Flip graph of subsets $G_{5,2}^{\mathsf{C}}$.

Figure 2 Examples of flip graphs with 1-rainbow cycles. In (d), two edge-disjoint rainbow Hamilton cycles in $G_{5,2}^{\mathsf{C}}$ are highlighted, one with bold edges and one with dashed edges.

interested in a cyclic enumeration of some of the combinatorial objects, such that each flip operation is encountered exactly once. For instance, in the flip graph of triangulations G_n^{T}, we ask for the existence of a cycle with the property that each inner edge of the triangulation appears (and disappears) exactly once. An example of such a cycle is shown in Figure 1. This idea can be formalized as follows. Consider two triangulations T and T' that differ in flipping the diagonal of a convex quadrilateral, i.e., T' is obtained from T by removing the diagonal e and inserting the other diagonal f. We view the edge between T and T' in the flip graph G_n^{T} as two arcs in opposite directions, where the arc from T to T' receives the label f, and the arc from T' to T receives the label e, so the label corresponds to the edge of the triangulation that enters in this flip; see the right hand side of Figure 1. Interpreting the

labels as colors, we are thus interested in a directed cycle in the flip graph in which each color appears exactly once, and we refer to such a cycle as a *rainbow cycle*. More generally, for any integer $r \geq 1$, an *r-rainbow cycle* in G_n^{T} is a cycle in which each edge of the triangulation appears (and disappears) exactly r times. Note that a rainbow cycle does not need to visit all vertices of the flip graph. Clearly, this notion of rainbow cycles extends in a natural way to all the other flip graphs discussed before, see Figure 2.

1.1 Our results

In this work we initiate the investigation of rainbow cycles in flip graphs for five popular classes of combinatorial objects. We consider three geometric classes: triangulations of a convex polygon, plane spanning trees on point sets in general position, and non-crossing perfect matchings on point sets in convex position. In addition, we consider two classes without geometric information: permutations of the set $[n] := \{1, 2, \ldots, n\}$, and k-element subsets of $[n]$. We proceed to present our results in these five settings in the order they were just mentioned. For the reader's convenience, all results are summarized in Table 1.

Our first result is that the flip graph of triangulations G_n^{T} defined in the introduction has a 1-rainbow cycle for $n \geq 4$ and a 2-rainbow cycle for $n \geq 7$ (Theorem 1 in Section 2).

Next, we consider the flip graph G_X^{S} of plane spanning trees on a point set X in general position; see Figure 2 (a). We prove that G_X^{S} has an r-rainbow cycle for any point set X with at least three points for any $r = 1, 2, \ldots, |X| - 2$ (Theorem 2 in Section 3).

We then consider the flip graph G_m^{M} of non-crossing perfect matchings on $2m$ points in convex position; see Figure 2 (b). We exhibit 1-rainbow cycles for $m = 2$ and $m = 4$ matching edges, and 2-rainbow cycles for $m = 6$ and $m = 8$. We argue that there is no 1-rainbow cycle

Table 1 Overview of results.

	Flip graph			Existence of r-rainbow cycle		
	Vertices	Arcs/edges	r	Yes	No	
GEOMETRIC	G_n^{T} triangulations of convex n-gon	edge flip	1	$n \geq 4$		Thm. 1
			2	$n \geq 7$		
	G_X^{S} plane spanning trees on point set X in general position	edge flip	$1, 2, \ldots, \lvert X\rvert - 2$	$\lvert X\rvert \geq 3$		Thm. 2
	G_m^{M} non-crossing perfect matchings on $2m$ points in convex position	two edge flip	1	$m \in \{2, 4\}$	odd m, $m \in \{6, 8, 10\}$	Thm. 3
			2	$m \in \{6, 8\}$		
ABSTRACT	G_n^{P} permutations of $[n]$	transposition	1	$\lfloor n/2 \rfloor$ even	$\lfloor n/2 \rfloor$ odd	Thm. 5
	$G_{n,k}^{\mathsf{C}}$ k-subsets of $[n]$, $2 \leq k \leq \lfloor n/2 \rfloor$	element exchange	1	odd n and $k < n/3$	even n	Thm. 6
	2-subsets of $[n]$ for odd n		1	two edge-disj. 1-rainbow Ham. cycles		

for $m \in \{6, 8, 10\}$, and none for odd m. In fact, we believe that there are no 1-rainbow cycles in G_m^{M} for any $m \geq 5$. Our results for this setting are summarized in Theorem 3 in Section 4.

Next, we consider the flip graph G_n^{P} of permutations of $[n]$, where an edge connects any two permutations that differ in a transposition, i.e., in exchanging two elements at positions i and j; see Figure 2 (c). The edges of this graph are colored with the corresponding pairs $\{i, j\}$, and in a 1-rainbow cycle each of the $\binom{n}{2}$ possible pairs appears exactly once. We prove that G_n^{P} has a 1-rainbow cycle if $\lfloor n/2 \rfloor$ is even, and no 1-rainbow cycle if $\lfloor n/2 \rfloor$ is odd (Theorem 5 in Section 5).

Finally, we consider the flip graph $G_{n,k}^{\mathsf{C}}$ of k-element subsets of $[n]$, also known as (n, k)-combinations, where an edge connects any two subsets that differ in exchanging one element i for another element j; see Figure 2 (d). The edges of this graph are colored with the corresponding pairs $\{i, j\}$, and in a 1-rainbow cycle each of the $\binom{n}{2}$ possible pairs appears exactly once. Since $G_{n,k}^{\mathsf{C}}$ is isomorphic to $G_{n,n-k}^{\mathsf{C}}$ including the edge-coloring, we assume without loss of generality that $2 \leq k \leq \lfloor n/2 \rfloor$. We prove that G_n^{C} has a 1-rainbow cycle for every odd n and $k < n/3$, and we prove that it has no 1-rainbow cycle for any even n. The case $k = 2$ is of particular interest since a 1-rainbow cycle in the flip graph $G_{n,2}^{\mathsf{C}}$ is also a Hamilton cycle. Moreover, we show that $G_{n,2}^{\mathsf{C}}$ has in fact two edge-disjoint 1-rainbow Hamilton cycles. Our results in this setting are summarized in Theorem 6 in Section 6.

We conclude in Section 7 with some open problems.

1.2 Related work

The *binary reflected Gray code* is a classical algorithm to generate all 2^n bitstrings of length n by flipping a single bit in each step; see [21, 36]. Since its invention, binary Gray codes satisfying various additional constraints have been constructed; see [32]. Specifically, a Gray code with the property that the bit-flip counts in each of the n coordinates are balanced, i.e., they differ by at most 2, was first described by Tootill [34]; see also [6]. When n is a power of two, every bit appears and disappears exactly $1/2 \cdot 2^n/n =: r$ many times. This balanced Gray code therefore corresponds to an r-rainbow cycle in the corresponding flip graph. In this light, our results are a first step towards balanced Gray codes for other combinatorial classes. For 2-element subsets, we indeed construct perfectly balanced Gray codes.

The Steinhaus-Johnson-Trotter algorithm [20, 35], also known as 'plain changes', is a method to generate all permutations of $[n]$ by adjacent transpositions $i \leftrightarrow i + 1$. More generally, it was shown in [22] that all permutations of $[n]$ can be generated by any set of transpositions that form a spanning tree on the set of positions $[n]$; see also [31].

The generation of (n, k)-combinations subject to certain restrictions on admissible exchanges $i \leftrightarrow j$ has been studied widely. Specifically, it was shown that all (n, k)-combinations can be generated with only allowing exchanges of the form $i \leftrightarrow i + 1$ [8, 10, 30], provided that n is even and k is odd, or $k \in \{0, 1, n - 1, n\}$. The infamous *middle levels conjecture* asserts that all $(2k, k)$-combinations can be generated with only exchanges of the form $1 \leftrightarrow i$, and this conjecture has recently been proved in [14, 26].

Rainbow cycles and paths have also been studied in graphs other than flip graphs. A well-known conjecture in this context due to Andersen [4] asserts that every properly edge-colored complete graph on n vertices has a rainbow path of length $n - 2$, i.e., a path that has distinct colors along each of its edges. Progress towards resolving this conjecture was recently made by Alon, Pokrovskiy and Sudakov [2], and Balogh and Molla [5].

1.3 Outline of this paper

In the remainder of this paper, we present our results discussed before and summarized in Table 1 in the same order. Due to space limitations, we only present particularly illuminating proofs and also provide proof sketches. Full proofs can be found in the preprint version of this paper [13].

2 Triangulations

In this section we consider a convex n-gon on points labeled clockwise by $1, 2, \dots, n$, and we denote by $\mathcal{T}_n$ the set of all triangulations on these points. The graph G_n^{T} has $\mathcal{T}_n$ as its vertex set, and an arc (T, T') between two triangulations T and T' that differ in exchanging the diagonal $e \in T$ of a convex quadrilateral formed by two triangles for the other diagonal $f \in T'$; see Figure 1. We refer to this operation as a *flip*, and we denote it by (e, f). Furthermore, we label the arc (T, T') with the edge f, i.e., the edge that enters the triangulation in this flip. The set of arc labels of G_n^{T} is clearly $E_n := \{\{i, j\} \mid j - i > 1\} \setminus \{1, n\}$. Recall that we think of these labels as colors. An r-rainbow cycle in G_n^{T} is a directed cycle along which every label from E_n appears exactly r times. Clearly, the length of an r-rainbow cycle equals $r|E_n| = r(\binom{n}{2} - n)$. Given an r-rainbow cycle, the cycle obtained by reversing the orientation of all arcs is also an r-rainbow cycle since every edge that appears r times also disappears r times. The following theorem summarizes the results of this setting.

▶ **Theorem 1.** *The flip graph of triangulations G_n^{T} has the following properties:*
(i) *If $n \geq 4$, then G_n^{T} has a 1-rainbow cycle.*
(ii) *If $n \geq 7$, then G_n^{T} has a 2-rainbow cycle.*

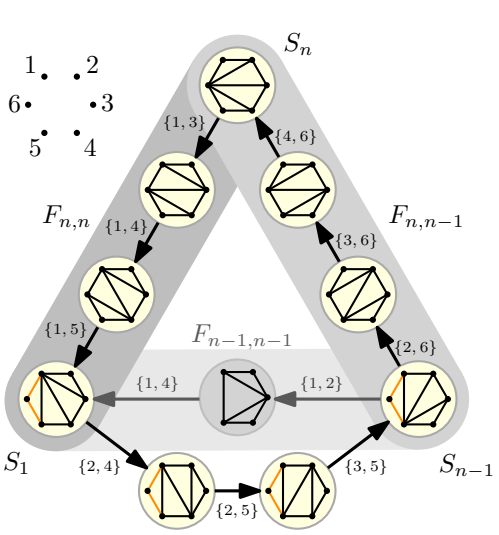
Proof. Let S_i be the star triangulation with respect to the point i, i.e., the triangulation where the point i has degree $n - 1$. To transform S_1 into S_2 we can use the flip sequence

$$F_{n,1} := \big((\{1,3\},\{2,4\}),(\{1,4\},\{2,5\}),(\{1,5\},\{2,6\}),\dots,(\{1,n-1\},\{2,n\})\big). \tag{1}$$

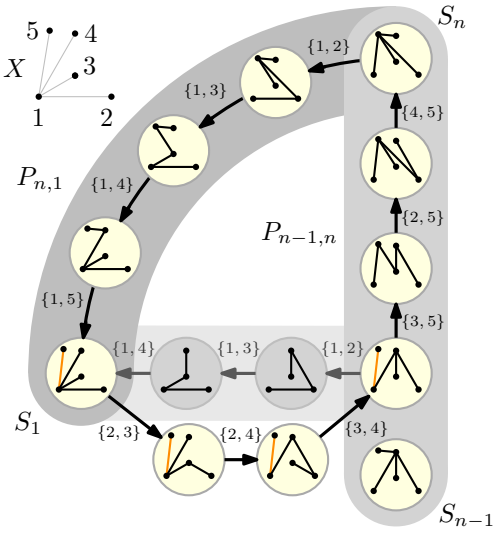
For any $i = 1, 2, \dots, n$, let $F_{n,i}$ denote the flip sequence obtained from $F_{n,1}$ by adding $i - 1$ to all points on the right-hand side of (1). Here and throughout this proof addition is to be understood modulo n with $\{1, 2, \dots, n\}$ as representatives for the residue classes. Note that $F_{n,i}$ transforms S_i into S_{i+1} for any $i \in [n]$, and all the edges from E_n that are incident with the point $i + 1$ appear exactly once during that flip sequence. Note also that $F_{n,i}$ has length $n - 3$.

We begin by proving (ii). The concatenation $(F_{n,1}, F_{n,2}, \dots, F_{n,n})$ is a flip sequence which applied to S_1 leads back to S_1. Along the corresponding cycle C in G_n^{T}, every edge from E_n appears exactly twice. Specifically, every edge $\{i, j\} \in E_n$ appears in the flip sequences $F_{n,i-1}$ and $F_{n,j-1}$. It remains to show that C is indeed a cycle, i.e., every triangulation appears at most once. For this observe that when applying $F_{n,i}$ to S_i, then for every $j = 1, 2, \dots, n - 4$, in the j-th triangulation we encounter after S_i, the point i is incident with exactly $n - 3 - j$ diagonals, the point $i + 1$ is incident with exactly j diagonals, while all other points are incident with at most two diagonals. Consequently, if $\min_{1 \leq j \leq n-4}\{n - 3 - j, j\} \geq 3$, then we can reconstruct uniquely, for any given triangulation encountered along C, which flip sequence $F_{n,i}$ it belongs to. This condition is satisfied if $n \geq 8$. For $n = 7$ it can be verified directly that C is a 2-rainbow cycle.

It remains to prove (i). We claim that for any $n \geq 4$, applying the flip sequence $X_n := (F_{4,3}, F_{5,4}, F_{6,5}, \dots, F_{n-1,n-2}, F_{n,n-1}, F_{n,n})$ to S_1 yields a 1-rainbow cycle in G_n^{T}.



(a) Illustration of the proof of Theorem 1 (i). In the induction step, the rainbow cycle on 5 points is extended to a rainbow cycle on 6 points.



(b) Illustration of the proof of Theorem 2 (i). In the induction step, the rainbow cycle from Figure 2 (a) on the point set [4] is extended to a rainbow cycle on the point set $X = [5]$.

Figure 3

Note that X_n differs from X_{n-1} by removing the terminal subsequence $F_{n-1,n-1}$, and by appending $F_{n,n-1}$ and $F_{n,n}$ to the shortened sequence, yielding a sequence of length $\left(\binom{n-1}{2} - (n-1)\right) - (n-4) + 2(n-3) = \binom{n}{2} - n$; see Figure 3 (a). The fact that X_n produces a rainbow cycle follows by induction, by observing that applying X_{n-1} to S_1 in G_n^{T} yields a cycle along which every edge from E_{n-1} appears exactly once. Moreover, along this cycle the point n is not incident with any diagonals. The modifications described before to construct X_n from X_{n-1} shorten this cycle in G_n^{T} and extend it by a detour through triangulations where the point n is incident with at least one diagonal. The set of edges that appear along this cycle is given by $(E_{n-1} \setminus E(F_{n-1,n-1})) \cup E(F_{n,n-1}) \cup E(F_{n,n-1}) = E_n$, where $E(F)$ denotes the set of edges appearing in a flip sequence F. This shows that applying X_n to S_1 yields a 1-rainbow cycle in G_n^{T}. ◀

3 Spanning trees

In this section we consider plane spanning trees on a set X of n points in general position, i.e., no three points are collinear. The graph G_X^{S} has the vertex set $\mathcal{S}_X$ consisting of all plane spanning trees on X, and an arc (T, T') between two spanning trees T and T' that differ in replacing an edge $e \in T$ by another edge $f \in T'$; see Figure 2 (a). We denote this *flip* by (e, f) and label the arc (T, T') with the edge f, i.e., the edge that enters the tree in this flip. Note that the entering edge f alone does not determine the flip uniquely (unlike for triangulations). Clearly, none of the two edges e and f can cross any of the edges in $T \cap T'$, but they may cross each other. The set of arc labels of G_X^{S} is clearly $E_X := \binom{X}{2}$. An r-rainbow cycle in G_X^{S} is a directed cycle along which every label from E_X appears exactly r times, so it has length $r\binom{n}{2}$. In the following theorem we summarize the results of this setting.

▶ **Theorem 2.** *The flip graph of plane spanning trees $G^{\mathtt{S}}_X$ has the following properties:*

(i) *For a point set X with $|X| \geq 3$ in general position, $G^{\mathtt{S}}_X$ has a 1-rainbow cycle.*

(ii) *For a point set X with $|X| \geq 4$ in general position and any $r = 2, 3, \ldots, m$, where $m := |X| - 1$ if $|X|$ is odd and $m := |X| - 2$ if $|X|$ is even, $G^{\mathtt{S}}_X$ has an r-rainbow cycle.*

Proof sketch. We first label an arbitrary point on the convex hull of X as point 1, and then label the points from 2 to n in counter-clockwise order around 1. Moreover, we denote by S_i the spanning tree that forms a star with center point i. We then define specific flip sequences that transform any star S_i into any other star S_j, yielding paths $P_{i,j}$ in the graph $G^{\mathtt{S}}_X$.

For proving (i), we use a similar inductive construction as for the proof of Theorem 1 (i), based on a strengthened induction hypothesis involving the path $P_{n,1}$; see Figure 3 (b).

To prove (ii) for even r, we use a decomposition of K_n, the complete graph on n vertices, into $r/2$ Hamilton cycles. Such a decomposition exists by Walecki's theorem [3]. Orienting each of the $r/2$ Hamilton cycles cyclically, we obtain a directed graph where each vertex has in- and out-degree $r/2$. We fix an Eulerian cycle $\mathcal{E}$ in this graph, and for each arc $(i, j) \in \mathcal{E}$ we add the path $P_{i,j}$ from S_i to S_j to obtain a directed closed tour C' in $G^{\mathtt{S}}_X$. In this tour every edge $\{i, j\} \in E_X$ appears exactly r times, $r/2$ times in every path towards S_i and $r/2$ times in every path towards S_j. However, the tour C' is not a cycle since every star S_i occurs $r/2$ times. We therefore bypass most of the stars by taking suitable detours. Each detour tree is obtained by exchanging the flip to the star with the flip from the star. This operation transforms C' into an r-rainbow cycle C.

For odd r, we perform the previous construction with $(r + 1)/2$ Hamilton cycles, and replace one by the 1-rainbow cycle constructed in part (i). ◀

4 Matchings

In this section we consider a set of $n = 2m$ points in convex position labeled clockwise by $1, 2, \ldots, n$. Without loss of generality we assume that the points are distributed equidistantly on a circle centered at the origin. $\mathcal{M}_m$ denotes the set of all non-crossing perfect matchings with m edges on these points. The graph $G^{\mathtt{M}}_m$ has $\mathcal{M}_m$ as its vertex set, and an arc (M, M') between two matchings M and M' that differ in exchanging two edges $e = \{a, b\} \in M$ and $f = \{c, d\} \in M$ for the edges $e' = \{a, c\}$ and $f' = \{b, d\} \in M'$; see Figure 2 (b). We refer to this operation as a *flip*, and label the arc (M, M') of $G^{\mathtt{M}}_n$ with the edges e' and f', i.e., the edges that enter the matching in this flip. The set of arc labels of $G^{\mathtt{M}}_m$ is $E_m := \{\{i, j\} \mid i, j \in [n] \text{ and } j - i \text{ is odd}\}$, and every arc of $G^{\mathtt{M}}_m$ carries two such labels. In this definition, the difference $j - i$ must be odd so that an even number of points lies on either side of the edge $\{i, j\}$. An r-rainbow cycle in $G^{\mathtt{M}}_m$ is a directed cycle along which every label in E_m appears exactly r times, two labels in each step, so it has length $r|E_m|/2 = rm^2/2$. The number of vertices of $G^{\mathtt{M}}_m$ is the Catalan number $\frac{1}{m+1}\binom{2m}{m}$. The following theorem summarizes the results of this setting.

▶ **Theorem 3.** *The flip graph of non-crossing perfect matchings $G^{\mathtt{M}}_m$, $m \geq 2$, has the following properties:*

(i) *If m is odd, then $G^{\mathtt{M}}_m$ has no 1-rainbow cycle.*

(ii) *If $m \in \{6, 8, 10\}$, then $G^{\mathtt{M}}_m$ has no 1-rainbow cycle.*

(iii) *If $m \in \{2, 4\}$, $G^{\mathtt{M}}_m$ has a 1-rainbow cycle, and if $m \in \{6, 8\}$, $G^{\mathtt{M}}_m$ has a 2-rainbow cycle.*

Proof of (i). A 1-rainbow cycle has length $m^2/2$, which is not integral for odd m. ◀

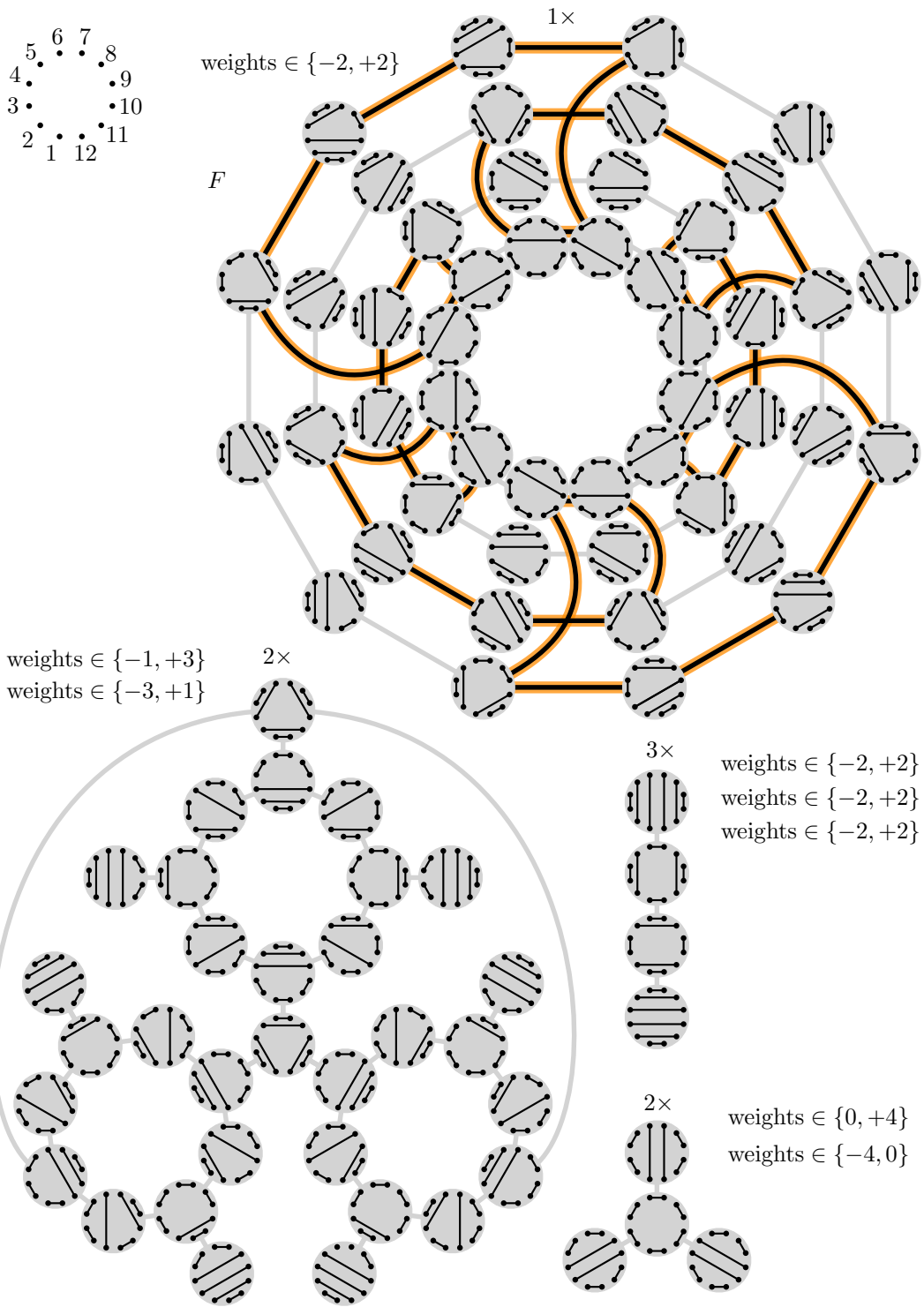


Figure 4 Illustration of the graph $H_6 \subseteq G_6^{\mathtt{M}}$. Some components of this graph are isomorphic to each other and differ only by rotation of the matchings by multiples of $\pi/6$. Only one representative for each component is shown, together with its multiplicity. The total number of matchings is the 6th Catalan number 132. The 2-rainbow cycle constructed in the proof of Theorem 3 (iii) is highlighted in the component F.

The *length* of any edge $e \in E_m$, denoted by $\ell(e)$, is the minimum number of points that lie on either of its two sides, divided by two. Consequently, a matching edge on the convex hull has length 0, whereas the maximum possible length is $(m-2)/2$, so there are $m/2$ different edge lengths. A convex quadrilateral formed by four edges from E_m is a *centered 4-gon*, if the sum of the edge lengths of the quadrilateral is $m-2$. Note that this is the maximum possible value. We refer to a flip involving a centered 4-gon as a *centered flip*. The following observation is crucial for our proof of Theorem 4 (ii).

▶ **Lemma 4.** *All flips along an r-rainbow cycle in $G_m^{\mathtt{M}}$ must be centered flips.*

Proof. E_m contains exactly $n = 2m$ edges of each length $0, 1, \ldots, (m-2)/2$. Along an r-rainbow cycle C, exactly rn edges of each length appear and disappear. Consequently, the average length of all edges that appear or disappear along C is $(m-2)/4$. By definition, in a centered flip the average length of the four edges involved in the flip is exactly the same number; whereas for a non-centered flip, it is strictly smaller. Therefore, C must not contain any non-centered flips. ◀

Lemma 4 allows to restrict our search for rainbow cycles to the subgraph H_m of $G_m^{\mathtt{M}}$ obtained by considering arcs that represent centered flips; consider Figure 4 for an example.

Proof sketch of (ii). To prove the case $m = 6$, we analyze each of the connected components of H_6 separately. Only one of them contains cycles of length $m^2/2 = 18$. This component is denoted by F in Figure 4. Analyzing the flip types along the arcs of this component, we show the non-existence of a 1-rainbow cycle in F. The proof for the cases $m = 8$ and $m = 10$ is computer-based, and uses exhaustive search for a 1-rainbow cycle in each connected component of H_m. ◀

Proof sketch of (iii). There are two non-crossing matchings with $m = 2$ edges, connected by two arcs in $G_m^{\mathtt{M}}$ that form 1-rainbow cycle. For $m = 4$ a 1-rainbow cycle is shown in Figure 2 (b). A 2-rainbow cycle for $m = 6$ is shown in Figure 4. A 2-rainbow cycle for $m = 8$ can be constructed using the path P of length 8 between matchings M and M' depicted in Figure 5. Note that M' differs from M by a counter-clockwise rotation by an angle of $\alpha := 2\pi/8$. Repeating this flip sequence eight times, rotating all flips by an angle of $\alpha \cdot i$ for $i = 0, 1, \ldots, 7$, yields a 2-rainbow cycle in $G_m^{\mathtt{M}}$. ◀

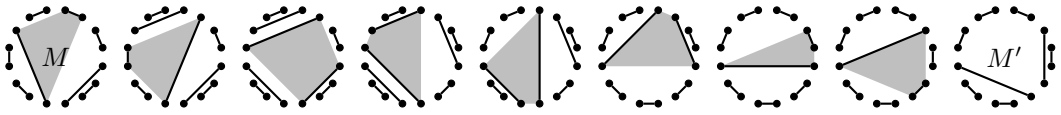


Figure 5 Definition of path P between matchings M and M' in $G_8^{\mathtt{M}}$.

5 Permutations

In this section, we consider the set of permutations Π_n of $[n]$. The graph $G_n^{\mathtt{P}}$ has vertex set Π_n, and an edge $\{\pi, \rho\}$ between any two permutations π and ρ that differ in exactly one transposition between the entries at positions i and j; see Figure 2 (c). We label the edge $\{\pi, \rho\}$ of $G_n^{\mathtt{P}}$ with the transposition $\{i, j\}$. A 1-rainbow cycle in $G_n^{\mathtt{P}}$ is a cycle of length $\binom{n}{2}$ along which every transposition appears exactly once. The following theorem summarizes the results in this setting.

▶ **Theorem 5.** *The flip graph of permutations G_n^{P}, $n \geq 2$, has the following properties:*

(i) *If $\lfloor n/2 \rfloor$ is odd, then G_n^{P} has no 1-rainbow cycle.*

(ii) *If $\lfloor n/2 \rfloor$ is even, then G_n^{P} has a 1-rainbow cycle.*

Proof sketch. The graph G_n^{P} is bipartite, where the partition classes are given by the parity of the number of inversions. If follows that a cycle of length $\binom{n}{2}$ cannot exist when this number is odd, which happens exactly when $\lfloor n/2 \rfloor$ is odd. This proves (i).

To prove (ii), observe that the graph G_n^{P} is vertex-transitive, so it suffices to specify a sequence of $\binom{n}{2}$ transpositions that yields a rainbow cycle. We refer to such a sequence as a *rainbow sequence for* Π_n. For the induction base $n = 4$, we use the rainbow sequence $R_4 := \big(\{1,2\},\{3,4\},\{2,3\},\{1,4\},\{2,4\},\{1,3\}\big)$. Applying R_4 to the permutation 1234 yields the rainbow cycle depicted in Figure 2 (c).

For the induction step, we assume that $n = 4\ell$ for some integer $\ell \geq 1$. From a rainbow sequence R_n for Π_n, we construct two new rainbow sequences, one for Π_{n+1} and another one for Π_{n+4}. This covers all values for which $\lfloor n/2 \rfloor$ is even. The inductive construction is based on the following trick: replacing a transposition $\{i,j\}$ in R_n by the sequence of three transpositions $(\{i,n+1\},\{i,j\},\{j,n+1\})$ has the same effect on the entries of the permutation, namely, swapping only the entries i and j and leaving the entry $n+1$ at the same position. However, in the modified sequence the transpositions $\{i,n+1\}$ and $\{j,n+1\}$ are used additionally. ◀

6 Subsets

In this section we consider the set of all k-element subsets of $[n]$, denoted by $C_{n,k} := \binom{[n]}{k}$, also called (n,k)-combinations. The graph $G_{n,k}^{\mathsf{C}}$ has vertex set $C_{n,k}$, and an edge $\{A,B\}$ between two sets A and B that differ in exchanging an element x for another element y, i.e., $A \setminus B = \{x\}$ and $B \setminus A = \{y\}$; see Figure 2 (d). We label the edge $\{A,B\}$ of $G_{n,k}^{\mathsf{C}}$ with the transposition $A \triangle B = \{x,y\} \in C_{n,2}$. A 1-rainbow cycle in $G_{n,k}^{\mathsf{C}}$ is a cycle of length $\binom{n}{2}$ where every transposition appears exactly once. We only consider 1-rainbow cycles in this setting, and we simply refer to them as rainbow cycles. Note that the number of vertices of $G_{n,k}^{\mathsf{C}}$ is $\binom{n}{k}$. Consequently, a rainbow cycle for $k = 2$ is in fact a Hamilton cycle, i.e., a Gray code in the classical sense. Since $G_{n,k}^{\mathsf{C}}$ and $G_{n,n-k}^{\mathsf{C}}$ are isomorphic, including the edge labels, we may assume without loss of generality that $k \leq n/2$. Moreover, observe that for $k = 1$ and $n > 3$, the length of the rainbow cycle exceeds the number of vertices of $G_{n,k}^{\mathsf{C}}$, so we also assume that $k \geq 2$. The following theorem summarizes the results in this setting.

▶ **Theorem 6.** *Let $n \geq 4$ and $2 \leq k \leq \lfloor n/2 \rfloor$. The flip graph of subsets $G_{n,k}^{\mathsf{C}}$ has the following properties:*

(i) *If n is even, there is no rainbow cycle in $G_{n,k}^{\mathsf{C}}$.*

(ii) *If n is odd and $k = 2$, then $G_{n,2}^{\mathsf{C}}$ has a rainbow Hamilton cycle.*

(iii) *If n is odd and $k = 2$, then $G_{n,2}^{\mathsf{C}}$ has two edge-disjoint rainbow Hamilton cycles.*

(iv) *If n is odd and $3 \leq k < n/3$, then $G_{n,k}^{\mathsf{C}}$ has a rainbow cycle.*

Proof of Theorem 6 (i). A fixed element $x \in [n]$ is involved in $n-1$ transpositions. If such a transposition is applied to a set containing x, then the resulting set does not contain x, and vice versa. In a rainbow cycle we return to the starting set and use each of these transpositions exactly once, so $n-1$ must be even, or equivalently, n must be odd. ◀

To prove parts (ii)–(iv) of Theorem 6, we construct rainbow cycles using a *rainbow block*. To introduce this notion, we need some definitions. For a set $A \subseteq [n]$, $\sigma(A)$ denotes the set

B_i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	dist(B_i)	dist($B_i \triangle B_{i+1}$) $= \lvert d_i \rvert$
B_1	×																×	1	
B_2	×		×															2	3
B_3	×														×			3	5
B_4	×				×													4	$7 = \ell - 1$
B_5	×					×												5	1
B_6	×											×						6	$6 = \ell - 2$
B_7	×							×										7	4
B_8	×									×								8	2
$\sigma(B_1)$	×	×																	dist($B_\ell \triangle \sigma(B_1)$) $= 8 = \ell$

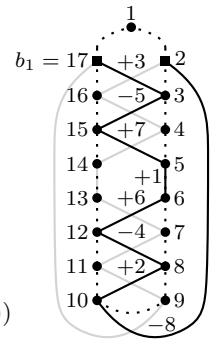


Figure 6 A rainbow block for $\ell = 8$. A cross in row B_i and column j indicates that $j \in B_i$. On the right, the sequence $(b_1, b_2, \ldots, b_\ell, 2)$ for this block is depicted as a path drawn in black. The path drawn in gray represents another rainbow block, yielding a rainbow Hamilton cycle that is edge-disjoint from the first cycle.

obtained from A by adding 1 to all elements, modulo n with $\{1, 2, \ldots, n\}$ as residue class representatives; for a pair $\{x, y\} \in C_{n,2}$, we define $\mathrm{dist}(\{x, y\}) := \min\{y - x, x - y\} \in [\ell]$ where the differences are taken modulo n. We call a sequence $B = (B_1, B_2, \ldots, B_\ell)$ with $B_i \in C_{n,k}$ a *rainbow block* if $C(B) := \big(B, \sigma^1(B), \sigma^2(B), \ldots, \sigma^{2\ell}(B)\big)$ is a rainbow cycle in $G^{\mathsf{C}}_{n,k}$. By definition, a rainbow cycle built from a rainbow block is highly symmetric.

Proof sketch of Theorem 6 (ii). Let $n = 2\ell + 1$ for some integer $\ell \geq 2$. We define a sequence $B = (B_1, B_2, \ldots, B_\ell)$ of pairs $B_i \in C_{n,2}$ such that the following conditions hold: (a) $B_i = \{1, b_i\}$ for $i \in [\ell]$ with $3 \leq b_i \leq n$ and $b_1 = n$, (b) $\{\mathrm{dist}(B_i) \mid i \in [\ell]\} = [\ell]$, and (c) $\{\mathrm{dist}(B_i \triangle B_{i+1}) \mid i \in [\ell - 1]\} \cup \{\mathrm{dist}(B_\ell \triangle \sigma(B_1)\} = [\ell]$. We claim that a sequence B satisfying these conditions yields a rainbow cycle $C = C(B)$. Indeed, (a) ensures that any two consecutive sets in C differ in exactly one transposition, and (b) and (c) guarantee that every pair $A \in C_{n,2}$ and every transposition $T \in C_{n,2}$, respectively, appear exactly once along C. An example of a rainbow block satisfying these conditions is shown in Figure 6.

It remains to construct a rainbow block with $B_i = \{1, b_i\}$ and $b_1 = n$. We define

$$d_i := \begin{cases} (-1)^{i+1} \cdot (2i+1) & \text{if } i \leq \lfloor (\ell-1)/2 \rfloor \\ (-1)^i & \text{if } i = \lfloor (\ell+1)/2 \rfloor \\ (-1)^{i+1} \cdot 2(\ell - i) & \text{if } i \geq \lfloor (\ell+2)/2 \rfloor \end{cases} \quad \text{and} \quad b_{i+1} := b_i + d_i \mod n \quad \text{for} \quad i \in [\ell - 1].$$

This definition satisfies conditions (a) and (b), and $\mathrm{dist}(B_i \triangle B_{i+1}) = |d_i|$ for all $i \in [\ell - 1]$. By definition, the set $\{|d_i| \mid i \in [\ell - 1]\}$ equals $[\ell] \setminus \{\ell\}$ if ℓ is even and it equals $[\ell] \setminus \{\ell - 1\}$ if ℓ is odd. These missing numbers are contributed by $\mathrm{dist}(B_\ell \triangle \sigma(B_1))$, so (c) is satisfied. ◀

The sequence $(b_1, b_2, \ldots, b_\ell, 2)$ can be interpreted as a path on the vertex set $[n]$; see the right part of Figure 6. The edge lengths of this path measured along the shorter of the two arcs along the cycle $(1, 2, \ldots, n)$ correspond to the $|d_i|$'s. We use this path representation to prove parts (iii) and (iv) of Theorem 6.

7 Open problems

- For all the combinatorial classes considered in this paper, it would be very interesting to exhibit r-rainbow cycles for larger values of r, in particular for the flip graphs of permutations and subsets. Another natural next step is to investigate rainbow cycles in

other flip graphs, e.g., for non-crossing partitions of a convex point set or for dissections of a convex polygon (see [17]).

- We believe that the flip graph of non-crossing perfect matchings $G_m^{\mathtt{M}}$ has no 1-rainbow cycle for any $m \geq 5$. This is open for the even values of $m \geq 12$. Moreover, the subgraph H_m of $G_m^{\mathtt{M}}$ restricted to centered flips (see Figure 4) is a very natural combinatorial object with many interesting properties that deserve further investigation. What is the number of connected components of H_m and what is their size? Which components are trees and which components contain cycles?
- We conjecture that the flip graph of subsets $G_{n,k}^{\mathtt{C}}$ has a 1-rainbow cycle for all $2 \leq k \leq n-2$. This is open for $n/3 \leq k \leq 2n/3$. In view of Theorem 6 (iii) we ask: does $G_{n,2}^{\mathtt{C}}$ have a factorization into $n-2$ edge-disjoint rainbow Hamilton cycles?

References

1. O. Aichholzer, F. Aurenhammer, C. Huemer, and B. Vogtenhuber. Gray code enumeration of plane straight-line graphs. *Graphs Combin.*, 23(5):467–479, 2007. doi:10.1007/s00373-007-0750-z.
2. N. Alon, A. Pokrovskiy, and B. Sudakov. Random subgraphs of properly edge-coloured complete graphs and long rainbow cycles. *arXiv:1608.07028*, Aug 2016.
3. B. Alspach. The wonderful Walecki construction. *Bull. Inst. Combin. Appl.*, 52:7–20, 2008.
4. L. D. Andersen. Hamilton circuits with many colours in properly edge-coloured complete graphs. *Mathematica Scandinavica*, 64:5–14, 1989.
5. J. Balogh and T. Molla. Long rainbow cycles and Hamiltonian cycles using many colors in properly edge-colored complete graphs. *arXiv:1706.04950*, Jun 2017.
6. G. S. Bhat and C. D. Savage. Balanced Gray codes. *Electron. J. Combin.*, 3(1):Research Paper 25, approx. 11 pp., 1996. URL: http://www.combinatorics.org/Volume_3/Abstracts/v3i1r25.html.
7. P. Bose and F. Hurtado. Flips in planar graphs. *Comput. Geom.*, 42(1):60–80, 2009. doi:10.1016/j.comgeo.2008.04.001.
8. M. Buck and D. Wiedemann. Gray codes with restricted density. *Discrete Math.*, 48(2-3):163–171, 1984. doi:10.1016/0012-365X(84)90179-1.
9. C. Ceballos, F. Santos, and G. M. Ziegler. Many non-equivalent realizations of the associahedron. *Combinatorica*, 35(5):513–551, 2015. doi:10.1007/s00493-014-2959-9.
10. P. Eades, M. Hickey, and R. C. Read. Some Hamilton paths and a minimal change algorithm. *J. Assoc. Comput. Mach.*, 31(1):19–29, 1984. doi:10.1145/2422.322413.
11. D. Eppstein. Happy endings for flip graphs. *J. Comput. Geom.*, 1(1):3–28, 2010.
12. R. Fabila-Monroy, D. Flores-Peñaloza, C. Huemer, F. Hurtado, J. Urrutia, and D. R. Wood. On the chromatic number of some flip graphs. *Discrete Math. Theor. Comput. Sci.*, 11(2):47–56, 2009.
13. S. Felsner, L. Kleist, T. Mütze, and L. Sering. Rainbow cycles in flip graphs. *arXiv:1712.07421*, Dec 2017.
14. P. Gregor, T. Mütze, and J. Nummenpalo. A short proof of the middle levels theorem. *arXiv:1710.08249*, Oct 2017.
15. S. Hanke, T. Ottmann, and S. Schuierer. The edge-flipping distance of triangulations. *Journal of Universal Computer Science*, 2(8):570–579, 1996.
16. C. Hernando, F. Hurtado, and M. Noy. Graphs of non-crossing perfect matchings. *Graphs Combin.*, 18(3):517–532, 2002. doi:10.1007/s003730200038.
17. C. Huemer, F. Hurtado, M. Noy, and E. Omaña-Pulido. Gray codes for non-crossing partitions and dissections of a convex polygon. *Discrete Appl. Math.*, 157(7):1509–1520, 2009. doi:10.1016/j.dam.2008.06.018.

18 F. Hurtado and M. Noy. Graph of triangulations of a convex polygon and tree of triangulations. *Comput. Geom.*, 13(3):179–188, 1999. doi:10.1016/S0925-7721(99)00016-4.

19 F. Hurtado, M. Noy, and J. Urrutia. Flipping edges in triangulations. *Discrete Comput. Geom.*, 22(3):333–346, 1999. doi:10.1007/PL00009464.

20 S. Johnson. Generation of permutations by adjacent transposition. *Math. Comp.*, 17:282–285, 1963.

21 D. E. Knuth. *The art of computer programming. Vol. 4A. Combinatorial algorithms. Part 1.* Addison-Wesley, Upper Saddle River, NJ, 2011.

22 V. L. Kompel'makher and V. A. Liskovets. Sequantial generation of arrangements by means of a basis of transpositions. *Kibernetica*, 3:17–21, 1975.

23 C. W. Lee. The associahedron and triangulations of the n-gon. *European J. Combin.*, 10(6):551–560, 1989. doi:10.1016/S0195-6698(89)80072-1.

24 M. Li and L. Zhang. Better approximation of diagonal-flip transformation and rotation transformation. In *Computing and Combinatorics, 4th Annual International Conference, COCOON '98, Taipei, Taiwan, R.o.C., August 12-14, 1998, Proceedings*, pages 85–94, 1998. doi:10.1007/3-540-68535-9_12.

25 J. M. Lucas. The rotation graph of binary trees is Hamiltonian. *J. Algorithms*, 8(4):503–535, 1987. doi:10.1016/0196-6774(87)90048-4.

26 T. Mütze. Proof of the middle levels conjecture. *Proc. Lond. Math. Soc. (3)*, 112(4):677–713, 2016. doi:10.1112/plms/pdw004.

27 A. Nijenhuis and H. S. Wilf. *Combinatorial algorithms.* Academic Press, Inc. [Harcourt Brace Jovanovich, Publishers], New York-London, second edition, 1978. For computers and calculators, Computer Science and Applied Mathematics.

28 L. Pournin. The diameter of associahedra. *Adv. Math.*, 259:13–42, 2014. doi:10.1016/j.aim.2014.02.035.

29 R. O. Rogers. On finding shortest paths in the rotation graph of binary trees. In *Proceedings of the Thirtieth Southeastern International Conference on Combinatorics, Graph Theory, and Computing (Boca Raton, FL, 1999)*, volume 137, pages 77–95, 1999.

30 F. Ruskey. Adjacent interchange generation of combinations. *J. Algorithms*, 9(2):162–180, 1988. doi:10.1016/0196-6774(88)90036-3.

31 F. Ruskey and C. Savage. Hamilton cycles that extend transposition matchings in Cayley graphs of S_n. *SIAM J. Discrete Math.*, 6(1):152–166, 1993. doi:10.1137/0406012.

32 C. Savage. A survey of combinatorial Gray codes. *SIAM Rev.*, 39(4):605–629, 1997. doi:10.1137/S0036144595295272.

33 D. D. Sleator, R. E. Tarjan, and W. P. Thurston. Rotation distance, triangulations, and hyperbolic geometry. *J. Amer. Math. Soc.*, 1(3):647–681, 1988. doi:10.2307/1990951.

34 G. C. Tootill. The use of cyclic permuted codes in relay counting circuits. *Proceedings of the IEE - Part B: Radio and Electronic Engineering*, 103:432–436, April 1956.

35 H. F. Trotter. Algorithm 115: Perm. *Commun. ACM*, 5(8):434–435, 1962. doi:10.1145/368637.368660.

36 H. S. Wilf. *Combinatorial algorithms: an update*, volume 55 of *CBMS-NSF Regional Conference Series in Applied Mathematics.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1989. doi:10.1137/1.9781611970166.

Hanani–Tutte for Approximating Maps of Graphs

Radoslav Fulek[1]
IST Austria
Am Campus 1, 3400 Klosterneuburg, Austria
radoslav.fulek@ist.ac.at
https://orcid.org/0000-0001-8485-1774

Jan Kynčl[2]
Department of Applied Mathematics and Institute for Theoretical Computer Science, Charles University, Faculty of Mathematics and Physics
Malostranské nám. 25, 118 00 Praha 1, Czech Republic
kyncl@kam.mff.cuni.cz
https://orcid.org/0000-0003-4908-4703

Abstract

We resolve in the affirmative conjectures of A. Skopenkov and Repovš (1998), and M. Skopenkov (2003) generalizing the classical Hanani–Tutte theorem to the setting of approximating maps of graphs on 2-dimensional surfaces by embeddings. Our proof of this result is constructive and almost immediately implies an efficient algorithm for testing whether a given piecewise linear map of a graph in a surface is approximable by an embedding. More precisely, an instance of this problem consists of (i) a graph G whose vertices are partitioned into clusters and whose inter-cluster edges are partitioned into bundles, and (ii) a region R of a 2-dimensional compact surface M given as the union of a set of pairwise disjoint discs corresponding to the clusters and a set of pairwise disjoint "pipes" corresponding to the bundles, connecting certain pairs of these discs. We are to decide whether G can be embedded inside M so that the vertices in every cluster are drawn in the corresponding disc, the edges in every bundle pass only through its corresponding pipe, and every edge crosses the boundary of each disc at most once.

2012 ACM Subject Classification Theory of computation → Computational geometry, Theory of computation → Graph algorithms analysis

Keywords and phrases Hanani–Tutte theorem, graph embedding, map approximation, weak embedding, clustered planarity

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.39

Related Version A full version of this paper is available at `https://arxiv.org/abs/1705.05243`.

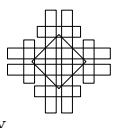
Acknowledgements We are grateful to Arkadiy Skopenkov for informing us about [30], and anonymous referees for comments that helped us to improve the presentation of the results.

1 Introduction

The Hanani–Tutte theorem is a classical result [20, 32] stating that a graph G is planar if it can be drawn in the plane so that every pair of edges not sharing a vertex cross an

[1] The author gratefully acknowledges support from Austrian Science Fund (FWF): M2281-N35.
[2] Supported by project 16-01602Y of the Czech Science Foundation (GAČR) and by Charles University project UNCE/SCI/004.

34th International Symposium on Computational Geometry (SoCG 2018).
Editors: Bettina Speckmann and Csaba D. Tóth; Article No. 39; pp. 39:1–39:15



Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

even number of times. According to Schaefer [29, Remark 3.6], *"The planarity criterion of Hanani–Tutte brings together computational, algebraic and combinatorial aspects of the planarity problem."* Perhaps the most remarkable algorithmic aspect of this theorem is that it implies the existence of a polynomial-time algorithm for planarity testing [28, Section 1.4.2]. In particular, the Hanani–Tutte theorem reduces planarity testing to solving a system of linear equations over $\mathbb{Z}_2$.

Seeing a graph G as a 1-dimensional topological space, an embeddability-testing algorithm decides whether there exists an injective continuous map, also called an *embedding*, $\psi : G \to M$, where M is a given triangulated compact 2-dimensional manifold without boundary. It is a classical result of Hopcroft and Tarjan that graph embeddability in the plane can be tested in linear time [22], and a linear-time algorithm is also known for testing whether G can be embedded into an arbitrary compact 2-dimensional manifold M [24], though computing the orientable genus (as well as Euler genus and non-orientable genus) of a graph is NP-hard [31]. We study a variant of this algorithmic problem in which we are given a piecewise linear continuous map $\varphi : G \to M$, which is typically not an embedding, and we are to decide whether for every $\varepsilon > 0$ there exists an embedding $\psi : G \to M$ such that $\|\psi - \varphi\| < \varepsilon$, where $\|.\|$ is the supremum norm. Such a map ψ is called an *ε-approximation* of φ, and in this case we say that φ is *approximable by an embedding*; or as in [2], a *weak embedding*. If φ is a constant map, the problem is clearly equivalent to the classical planarity testing. Obviously, an instance of our problem is negative if there exists a pair of edges e and g in G such that the curves $\varphi(e)$ and $\varphi(g)$ induced by φ properly cross. Hence, in a typical instance of our problem the map φ somewhat resembles an embedding except that we allow a pair of edges to overlap and an edge to be mapped onto a single point.

Building upon the work of Minc [23], M. Skopenkov [30] gave an algebraic characterization via van Kampen obstructions of maps φ approximable by an embedding in the plane in the case when G is a cycle or when G is subcubic and the image of φ is a simple closed curve. This implies a polynomial-time algorithm for the decision problem in the corresponding cases and can be seen as a variant of the characterization of planar graphs due to Hanani and Tutte. The aim of this work is to prove a conjecture of M. Skopenkov [30, Conjecture 1.6] generalizing his results along with its algorithmic consequences to arbitrary graphs. Independently of the aforementioned developments, a series of recent papers [1, 7, 9] on weakly simple embeddings shows that the problem of deciding the approximability of φ by an embedding is tractable and can be carried out in $O(|\varphi| \log |\varphi|)$ time, where $|\varphi|$ is the number of line segments specifying φ.

In spite of the analytic definition, the algorithmic problem of deciding whether φ is approximable by an embedding admits a polynomially equivalent reformulation that is of combinatorial flavor and that better captures the essence of the problem. Therefore we state our results in terms of the reformulation, whose planar case is a fairly general restricted version of the c-planarity problem [10, 11] of Feng, Cohen and Eades introduced by Cortese et al. [9]. The computational complexity of c-planarity testing is a well-known notoriously difficult open problem in the area of graph visualization [8]. To illustrate this state of affairs we mention that Angelini and Da Lozzo [5] have recently studied our restricted variant (as well as its generalizations) under the name of *c-planarity with embedded pipes* and provided an FPT algorithm for it [5, Corollary 18].

Roughly speaking, in the *clustered planarity* problem, shortly *c-planarity*, we are given a planar graph G equipped with a hierarchical structure of subsets of its vertex set. The subsets are called *clusters*, and two clusters are either disjoint or one contains the other. The question is whether a planar embedding of G with the following property exists: the vertices

in each cluster are drawn inside a disc corresponding to the cluster so that the boundaries of the discs do not intersect, the discs respect the hierarchy of the clusters, and every edge in the embedding crosses the boundary of each disc at most once.

Notation. Let us introduce the notation necessary for precisely stating the problem that we study. Let $G = (V, E)$ be a multigraph without loops. If we treat G as a topological space, then a *drawing* ψ of G is a piecewise linear map from G into a triangulated 2-dimensional manifold M where every vertex in $V(G)$ is mapped to a unique point and every edge $e \in E$ joining u and v is mapped bijectively to a simple arc joining $\psi(u)$ and $\psi(v)$. We understand E as a multiset, and by a slight abuse of notation we refer to an edge e joining u and v as uv even though there might be other edges joining the same pair of vertices. Multiple edges are mapped to distinct arcs meeting at their endpoints. Given a map m we denote by $m|_X$, where X is a subset of the domain of m, the function obtained from m by restricting its domain to X. If H is a graph equipped with an embedding, we denote by $H|_X$, where X is a subgraph of H, the graph X with the embedding inherited from H.

If it leads to no confusion, we do not distinguish between a vertex or an edge and its image in the drawing and we use the words "vertex" and "edge" in both contexts. Also when talking about a drawing we often mean its image.

We assume that drawings satisfy the following standard general position conditions. No edge passes through a vertex, every pair of edges intersect in finitely many points, no three edges intersect at the same inner point, and every intersection point between a pair of edges is realized either by a proper crossing or a common endpoint. Here, by a *proper crossing* we mean a transversal intersection that is a single point.

An *embedding* of a graph G is a drawing of G in M without crossings. The *rotation* at a vertex v in a drawing of G is the clockwise cyclic order of the edges incident to v in a small neighborhood of v in the drawing w.r.t a chosen orientation at the vertex. The *rotation system* of a drawing of G is the set of rotations of all the vertices in the drawing. The embedding of G is combinatorially determined by the rotation system and consistently chosen orientations at the vertices if M is orientable. If M is non-orientable we need to additionally specify the signs of the edges as follows. We assume that M is constructed from a punctured 2-sphere by turning all the holes into cross-caps, i.e., by identifying the pairs of opposite points on every hole. A *sign* on an edge is positive if overall the edge passes an even number of times through the cross-caps, and negative otherwise.

Refer to Figure 1. We refer the reader to the monograph by Mohar and Thomassen [25] for a detailed introduction into surfaces and graph embeddings. Let $\varphi : G \to M$ be a piecewise linear map with finitely many linear pieces. Suppose that φ is free of edge crossings, and in φ, edges do not pass through vertices. As we will see later, the image of φ can be naturally identified with a graph H embedded in M. Throughout the paper we denote both vertices and edges of H by Greek letters. Let the *thickening* $\mathcal{H}$ of H be a 2-dimensional surface with boundary obtained as a quotient space of a union of pairwise disjoint topological discs as follows. We take a union of pairwise disjoint closed discs $\mathcal{D}(\nu)$, called *clusters*, for all $\nu \in V(H)$ and closed rectangles $\mathcal{P}(\rho)$, called *pipes*, for all $\rho \in E(H)$. We connect every pair of discs $\mathcal{D}(\nu)$ and $\mathcal{D}(\mu)$, such that $\rho = \nu\mu \in E(H)$, by $\mathcal{P}(\rho)$ in correspondence with the rotations at vertices of the given embedding of H as described next. Let ∂X denote the boundary of X. We consider a subset of $\partial\mathcal{D}(\nu)$, for every $\nu \in V(H)$, consisting of $\deg(\nu)$ pairwise disjoint closed (non-trivial) arcs $\mathcal{A}(\nu, \mu)$, one for every $\nu\mu \in E(H)$, appearing along $\partial\mathcal{D}(\nu)$ in correspondence with the rotation of ν. For every $\mathcal{D}(\nu)$, we fix an orientation of $\partial\mathcal{D}(\nu)$ and $\partial\mathcal{P}(\nu\mu)$.

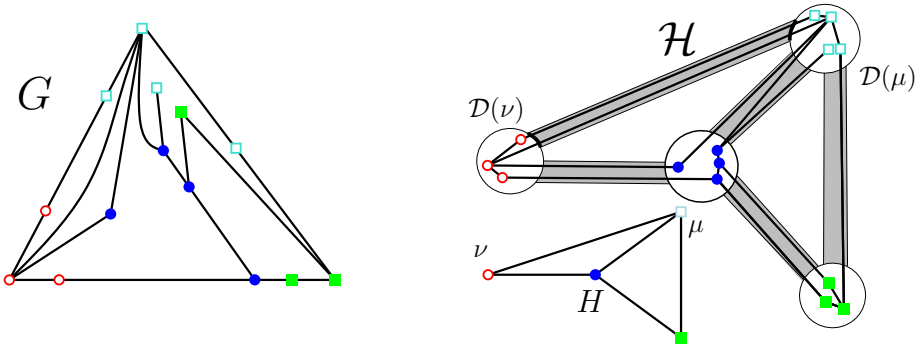


Figure 1 An instance (G, H, φ) and its approximation by an embedding contained in the thickening $\mathcal{H}$ of H. The valves of the pipe of $\rho = \nu\mu$ at $\mathcal{D}(\nu)$ and $\mathcal{D}(\mu)$ are highlighted by bold arcs.

If M is orientable, for every $\mathcal{P}(\nu\mu)$, we identify by an orientation reversing homeomorphism its opposite sides with $\mathcal{A}(\nu, \mu)$ and $\mathcal{A}(\mu, \nu)$ w.r.t. the chosen orientations of $\partial\mathcal{D}(\nu)$, $\partial\mathcal{D}(\mu)$, and $\partial\mathcal{P}(\nu\mu)$. If M is non-orientable, for every $\mathcal{P}(\nu\mu)$ with the positive sign we proceed as in the case when M is orientable and for every every $\mathcal{P}(\nu\mu)$ with the negative sign, we identify by an orientation preserving homeomorphism its opposite sides with $\mathcal{A}(\nu, \mu)$ and $\mathcal{A}(\mu, \nu)$ w.r.t. the chosen orientations of $\partial\mathcal{D}(\nu)$ and $\partial\mathcal{P}(\nu\mu)$, and the reversed orientation of $\partial\mathcal{D}(\mu)$. We call the intersection of $\partial\mathcal{D}(\nu) \cap \partial\mathcal{P}(\nu\mu)$ a *valve* of $\nu\mu$.

Instance. An instance of the problem that we study is defined as follows. The instance is a triple (G, H, φ) of an (abstract) graph G, a graph H embedded in a closed 2-dimensional manifold M, and a map $\varphi : V(G) \to V(H)$ such that every pair of vertices joined by an edge in G are mapped either to a pair of vertices joined by an edge in H or to the same vertex of H. We naturally extend the definition of φ to each subset U of $V(G)$ by putting $\varphi(U) = \{\varphi(u)|\ u \in U\}$, and to each subgraph G_0 of G by putting $\varphi(G_0) = (\varphi(V(G_0)), \{\varphi(e)|\ e \in E(G_0), |\varphi(e)| = 2\})$. The map φ induces a partition of the vertex set of G into *clusters* V_ν, where $V_\nu = \varphi^{-1}[\nu]$.

Question. Decide whether there exists an embedding ψ of G in the interior of a thickening $\mathcal{H}$ of H so that the following hold.

(A) Every vertex $v \in V_\nu$ is drawn in the interior of $\mathcal{D}(\nu)$, i.e., $\psi(v) \in \mathrm{int}(\mathcal{D}(\nu))$.
(B) For every $\nu \in V(H)$, every edge $e \in E(G)$ intersecting $\partial\mathcal{D}(\nu)$ does so in a single proper crossing, i.e., $|\psi(e) \cap \partial\mathcal{D}(\nu)| \leq 1$.

Note that conditions (A) and (B) imply that every edge of G is allowed to pass through at most one pipe as long as G is drawn in $\mathcal{H}$. The instance is *positive* if an embedding ψ of G satisfying (A) and (B) exists and *negative* otherwise. If (G, H, φ) is a positive instance we say that (G, H, φ) is *approximable by the embedding* ψ, shortly *approximable*. We call ψ the *approximation* of (G, H, φ). When the instance (G, H, φ) is clear from the context, we call ψ the *approximation* of φ.

The instance (G, H, φ), or shortly φ, is *locally injective* if for every vertex $v \in V(G)$, the restriction of φ to the union of v and the set of its neighbors is injective, or equivalently, no two vertices that are adjacent or have a common neighbor in G are mapped by φ to the same vertex in H. An edge of G is a *pipe edge* if it is mapped by φ to an edge of H. When talking about pipe edges, we have a particular instance in mind, which is clear from the context.

The *pipe degree*, $pdeg(C)$, of a subgraph C of $G[V_\nu]$ is the number of edges ρ of H for which there exists a pipe edge e with one vertex in C such that $\varphi(e) = \rho$.

1.1 The result

An edge in a drawing is *even* if it crosses every other edge an even number of times. A vertex in a drawing is *even* if every pair of its incident edges cross an even number of times. An edge in a drawing is *independently even* if it crosses every other non-adjacent edge an even number of times. A drawing of a graph is *(independently) even* if all edges are (independently) even. Note that every embedding is an even drawing.

We formulate our main theorem in terms of a relaxation of the notion of an approximable instance (G, H, φ). An instance (G, H, φ) is $\mathbb{Z}_2$*-approximable* if there exists an independently even drawing of G in the interior of $\mathcal{H}$ satisfying (A) and (B). We call such a drawing a $\mathbb{Z}_2$*-approximation* of (G, H, φ). The proof of the Hanani–Tutte theorem from [26] proves that given an independently even drawing of a graph in the plane, there exists an embedding of the graph in which the rotations at even vertices are preserved, that is, they are the same as in the original independently even drawing. We refer to this statement as to the *unified Hanani–Tutte theorem* [14]. Our result can be thought of as a generalization of this theorem, which also motivates the following definition. A drawing ψ of G is *compatible* with a drawing ψ_0 of G if every even vertex in ψ_0 is also even in ψ and has the same rotation in both drawings ψ_0 and ψ.[3]

It is known that $\mathbb{Z}_2$-approximability of (G, H, φ) does not have to imply its approximability by an embedding [27, Figure 1(a)]. Our main result characterizes the instances (G, H, φ) for which such implication holds. The characterization is formulated in terms of the derivative of (G, H, φ), whose formal definition is postponed to Section 2, since its definition relies on additional concepts that we need to introduce, which would unnecessarily further delay stating of our main result.

▶ **Theorem 1.** *If an instance (G, H, φ) is $\mathbb{Z}_2$-approximable by an independently even drawing ψ_0 then either (G, H, φ) is approximable by an embedding ψ compatible with ψ_0, or it is not approximable by an embedding and in the ith derivative $(G^{(i)}, H^{(i)}, \varphi^{(i)})$, for some $i \in \{1, 2, \ldots, 2|E(G)|\}$, there exists a connected component $C \subseteq G^{(i)}$ such that C is a cycle, $\varphi^{(i)}$ is locally injective and $(C, H^{(i)}|_{\varphi^{(i)}(C)}, \varphi^{(i)}|_C)$ is not approximable by an embedding.*

The obstruction $(C, H^{(i)}|_{\varphi^{(i)}(C)}, \varphi^{(i)}|_C)$ from the statement of the theorem has the form of the "standard winding example" [27, Figure 1(a)], in which the cycle C is forced by $\varphi^{(i)}$ to wind around a point inside a face of H more than once (and an odd number of times, since it has a $\mathbb{Z}_2$-approximation). Our main result implies the following.

▶ **Corollary 2.** *If G is a forest, the $\mathbb{Z}_2$-approximability implies approximability by an embedding.*

Theorem 1 confirms a conjecture of M. Skopenkov [30, Conjecture 1.6], since our definition of the derivative agrees with his definition in the case when G is a cycle and since for every cycle C in $G^{(i)}$ there exists a cycle D in G such that $(D^{(i)}, H^{(i)}|_{\varphi(D^{(i)})}, \varphi^{(i)}|_{D^{(i)}}) = (C, H^{(i)}|_{\varphi^{(i)}(C)}, \varphi^{(i)}|_C)$. Corollary 2 confirms a conjecture of Repovš and A. Skopenkov [27, Conjecture 1.8]. The main consequence of Theorem 1 is the following.

[3] Since in general we work also with non-orientable surfaces the rotation is determined only up to the choice of an orientation at each vertex for non-orientable surfaces.

▶ **Theorem 3.** *We can test in $O(|\varphi|^{2\omega})$, where $O(n^\omega)$ is the complexity of multiplication*[4] *of square $n \times n$ matrices, whether (G, H, φ) is approximable by an embedding.*

Theorem 3 implies tractability of c-planarity with embedded pipes [9] and therefore solves a related open problem of Chang, Erickson and Xu [7, Section 8.2] and Akitaya et al. [1]. The theorem also implies that strip planarity introduced by Angelini et al. [3] is tractable, and hence, solves the main problem asked therein. The theorem generalizes results of [13] and [17], and implies that c-planarity [10, 11] for flat clustered graphs is tractable for instances with three clusters, which has been open, to the best of our knowledge. We remark that only solutions to the problem for two clusters were given so far [6, 19, 21]. Nevertheless, after the completion of this work our running time was improved to $O(|\varphi|^2 \log |\varphi|)$ [2]. The improvement on the running time was achieved by using a similar strategy as in the present work, while eliminating the need to solve the linear system and employing a very careful running time analysis. Previously, polynomial running time $O(|\varphi|^4)$ was obtained by the first author [12] for graphs with fixed combinatorial embedding.

We mention that Theorem 1 and Theorem 3 easily generalize to the setting when clusters are homeomorphic to cylinders and $\mathcal{H}$ is homeomorphic to a torus or a cylinder, which extends some recent work [4, 15, 16]. It is an interesting open problem to find out if the technique of [2] generalizes to this setting as well.

Our proof of Theorem 1 extends the technique of Minc [23] and M. Skopenkov [30]. In particular, we extend the definition of the derivative for maps of graphs to instances in a certain normal form which can be assumed without loss of generality. Roughly, the derivative is an operator that takes an input instance (G, H, φ) and produces a simpler instance (G', H', φ') that is positive if and only (G, H, φ) is. We remark that the operations of *cluster expansion* and *pipe contraction* of Cortese et al. [9] bear many similarities with the derivative, and can be considered as local analogs of the derivative. One of the reasons for introducing the normal form is to impose on the instance conditions analogous to the properties of a *contractible base* [9], or a *safe pipe* [2], which make the derivative reversible.

Organization. In Section 2, we define the normal form of instances and the operation of the derivative for instances in the normal form. Furthermore, we state a claim (Claim 8) saying that for any instance admitting a $\mathbb{Z}_2$-approximation there exists, in some sense, an equivalent instance in the normal form. Thus, it is enough to define the derivative only for instances in the normal form. In Section 3, we prove Theorem 1.

2 Normal form and derivative

Normal form. We define the normal form of an instance (G, H, φ) to which we can apply the derivative. In order to keep the definition more compact we define the normal form via its topologically equivalent subdivided variant; see Figure 2 for an illustration. This variant also facilitates the definition of the derivative and those are its only purposes in this work. Roughly speaking, (G, H, φ) is in the subdivided normal form if there exists an independent set $V_s \subset V(G)$ without degree-1 vertices such that every connected component C of $G[V \setminus V_s]$ is mapped by φ to an edge $\varphi(C) = \nu\mu$ of H and both its parts mapped to ν and μ are forests.

[4] The best current algorithms for matrix multiplication give $\omega < 2.3729$ [18, 34]. Since a linear system appearing in the solution is sparse, it is also possible to use Wiedemann's randomized algorithm [33], with expected running time $O(|\varphi|^4 \log |\varphi|^2)$ in our case.

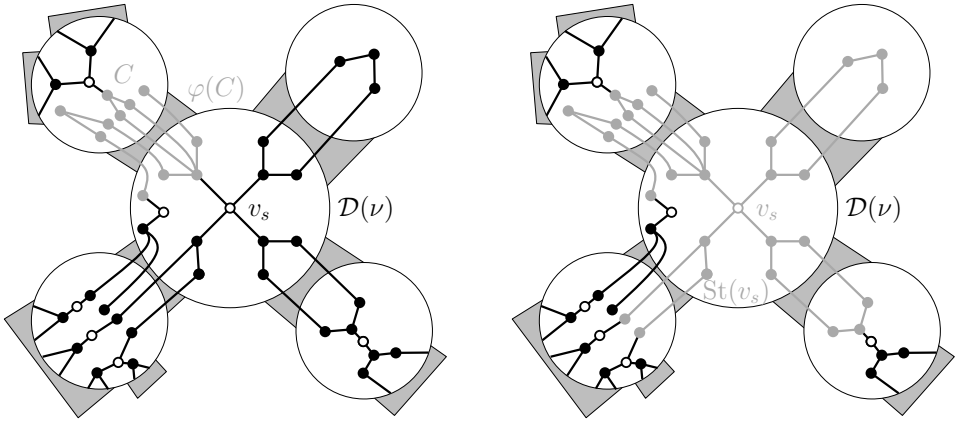


Figure 2 A part of (G, H, φ) in the subdivided normal form illustrated by its approximation in $\mathcal{H}$. A connected component C of $G[V \setminus V_s]$ (left) and $\mathrm{St}(v_s)$ (right), both colored gray. The empty vertices belong to the independent set $V_s \subset V(G)$.

We call vertices in V_s *central vertices*, which conveys an intuition that every vertex of V_s constitutes in some sense a center of a connected component induced by a cluster.

The normal form is obtained from the subdivided normal form by *suppressing* in V_s any vertices of degree 2, i.e., by replacing each such vertex v_s and both its incident edges by a single edge, and performing the same replacement for $\varphi(v_s)$ which must be also of degree 2; see Figure 3 left for an illustration.

▶ **Definition 4.** The instance (G, H, φ) is in the *subdivided normal form* if there exists an independent set $V_s \subset V(G)$ with the following properties.
For every connected component C of $G[V \setminus V_s]$ the image $\varphi(C)$ is an edge of $E(H)$; and $\varphi^{-1}|_\nu[\varphi(C)]$ is a forest for both vertices $\nu \in \varphi(C)$. For every connected component C of $G[V_\nu]$, for every $\nu \in V(H)$ the following holds. If $pdeg(C) \geq 2$ then $|V(C) \cap V_s| = 1$, and $V(C) \cap V_s = \emptyset$ otherwise; for every $v_s \in V_s \cap V(C)$ we have that $\deg(v_s) = pdeg(C)$; and no two edges incident to v_s join v_s with connected components C_1 and C_2 of $G[V \setminus V_s]$ such that $\varphi(C_1) = \varphi(C_2)$.

The instance obtained from an instance (G, H, φ) in the subdivided normal form by suppressing all degree-2 vertices in V_s is in the *normal form*. Such an instance in the normal form *corresponds* to the original instance (G, H, φ) in the subdivided normal form, and vice-versa.

Derivative. The rest of the section is inspired by the work of M. Skopenkov [30] and also Minc [23]. In particular, the notion of the derivative in the context of map approximations was introduced by Minc and adapted to the setting of $\mathbb{Z}_2$-approximations by M. Skopenkov for instances (G, H, φ) where G is subcubic and H is a cycle. We extend his definition to instances (G, H, φ) in the normal form. (A somewhat simplified extension was already used in [12].) Thus, by derivating (G, H, φ) we, in fact, mean bringing the instance (G, H, φ) into the normal form and then derivating the instance in the normal form. Given the instance (G, H, φ) in the normal form or the subdivided normal form already, the operation of the derivative outputs an instance (G', H', φ'), where $G' = G$ and H' is defined next.

In order to keep the definition more compact we formulate it first for the instances in the subdivided normal form. Thus, in the following we assume (G, H, φ) to be in the subdivided normal form.

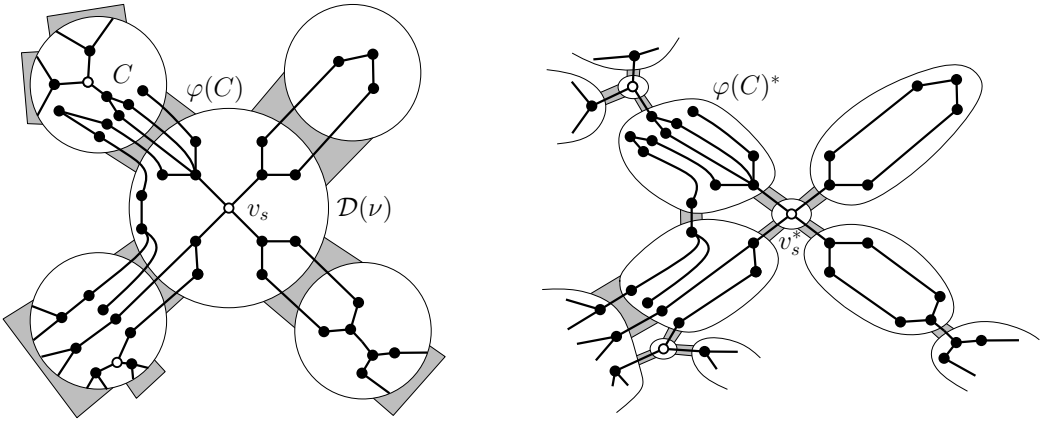


Figure 3 A part of (G, H, φ) in the normal form (left) illustrated by its approximation in $\mathcal{H}$, and its derivative (right). The empty vertices belong to the independent set $V_s \subset V(G)$.

Let V_s be the set of central vertices in G. Let $\mathcal{G}$ be a bipartite graph with the vertex set $V_s \cup \mathcal{C}$, where $\mathcal{C} = \{C |\ C \text{ is a connected component of } G[V \setminus V_s]\}$, in which $v_s \in V_s$ and $C \in \mathcal{C}$ are joined by an edge if and only if v_s is joined by an edge with a vertex of C in G.

Let the *star* of v_s, $\mathrm{St}(v_s)$, with $v_s \in V_s$ be the subgraph of G induced by $\{v_s\} \cup \bigcup_{C \in V(\mathcal{G}):\ v_s C \in E(\mathcal{G})} V(C)$; see Figure 2 (right) for an illustration. The vertices of H' are in the bijection with the union of V_s with the set of the edges of H. This bijection is denoted by superscript $*$. The graph H' is a bipartite graph with the vertex set $V(H') = \{\rho^* | \rho \in E(H)\} \cup \{v_s^* |\ v_s \in V_s\}$. We have $\rho^* v_s^* \in E(H')$ if and only if $\rho \in E(\varphi(\mathrm{St}(v_s)))$. We use the convention of denoting a vertex in $V(H')$ whose corresponding edge in $E(H)$ is $\rho = \nu\mu$ by both ρ^* or $(\nu\mu)^*$. An embedding of H' in M and signs on the edges of H', if M is non-orientable, are naturally inherited from those of H. Figure 3 (right) illustrates the restriction of the embedding of H' to its subgraph "stemming" from ν.

▶ **Definition 5.** Let (G, H, φ) be $\mathbb{Z}_2$-approximable and in the subdivided normal form. The *derivative* $(G, H, \varphi)'$ of (G, H, φ) is the instance (G', H', φ') such that $\varphi'(v_s) = v_s^*$, for $v_s \in V_s$, and $\varphi'(v) = \varphi(C)^*$, for every $v \in V(C)$, where C is a connected component of $G[V \setminus V_s]$. (Hence, $\varphi(C)$ is an edge of H by the definition of the subdivided normal form.)

The *derivative* $(G, H, \varphi)'$ of (G, H, φ), where (G, H, φ) is $\mathbb{Z}_2$-approximable and in the normal form, is the instance obtained from the derivative of the corresponding instance in the subdivided normal form by suppressing every vertex v_s of degree 2 in V_s and its image $\varphi(v_s)$ in H', and eliminating multiple edges in H'.

Remark. Since (G', H', φ') is defined only for instances in the (subdivided) normal form, by derivating an instance, which is not in the normal form, we will mean an operation that, first, brings the instance into the normal form, and second, applies the derivative to the instance. We take the liberty of denoting by G', H' and φ' an object that does not depend only on G, H and φ, respectively, but on the whole instance (G, H, φ).

The proof of Theorem 1 proceeds by induction on the *potential* $p(G, H, \varphi)$, which is always non-negative and is defined as follows. Let $E_p(G)$ be the set of pipe edges in G, we put $p(G, H, \varphi) = |E_p(G)| - |E(H)|$. We will show that an application of the derivative decreases the potential unless the instance is locally injective. The latter can be thought of as the base case of the induction. In order to prove that the inductive step goes through we will need the following three claims.

We show that if (G, H, φ) in the normal form is $\mathbb{Z}_2$-approximable then (G', H', φ') is $\mathbb{Z}_2$-approximable as well. More precisely, we prove the following claim.

▶ **Claim 6.** *If an instance (G, H, φ) in the normal form is $\mathbb{Z}_2$-approximable by a drawing ψ_0 then $(G' = G, H', \varphi')$ is $\mathbb{Z}_2$-approximable by a drawing ψ_0' such that ψ_0 is compatible with ψ_0'. Moreover, if ψ_0 is crossing free so is ψ_0'.*

The previous claim implies that if (G, H, φ) is approximable by an embedding the same holds for (G', H', φ'), which we use in the proof of Theorem 1 to conclude that if (G', H', φ') is not approximable by an embedding the same holds for (G, H, φ). However, we need also the converse of this to hold, which is indeed the case.

▶ **Claim 7.** *If the instance (G', H', φ') is approximable by an embedding ψ' compatible with ψ_0' from Claim 6 then (G, H, φ) is approximable by an embedding compatible with ψ_0.*

We say that $(\hat{G}, \hat{H}, \hat{\varphi}, \hat{\psi}_0)$ is a *clone* of (G, H, φ, ψ_0) if the following holds. If $(\hat{G}, \hat{H}, \hat{\varphi})$ is approximable by an embedding compatible with $\hat{\psi}_0$ then (G, H, φ) is approximable by an embedding compatible with ψ_0; and if (G, H, φ) is approximable by an embedding then $(\hat{G}, \hat{H}, \hat{\varphi})$ is approximable by an embedding. Note that being a clone is a transitive relation. However, the relation is not symmetric, and thus, it is not an equivalence relation.

Due to the following claim, it is indeed enough to work with instances in the normal form in Claim 6.

▶ **Claim 8.** *Given a $\mathbb{Z}_2$-approximation ψ_0 of (G, H, φ) there exist an instance $(\hat{G}, \hat{H}, \hat{\varphi})$ in the normal form that is $\mathbb{Z}_2$-approximable, such that $p(G, H, \varphi) = p(\hat{G}, \hat{H}, \hat{\varphi})$; and (2) a $\mathbb{Z}_2$-approximation $\hat{\psi}_0$ of $(\hat{G}, \hat{H}, \hat{\varphi})$ such that $(\hat{G}, \hat{H}, \hat{\varphi}, \hat{\psi}_0)$ is a clone of (G, H, φ, ψ_0).*

3 Proof of Theorem 1

Let (G, H, φ) be an instance that is $\mathbb{Z}_2$-approximable by an independently even drawing ψ_0. We start with a claim that helps us to identify instances that cannot be further simplified by derivating. We show that by successively applying the derivative we eventually obtain an instance such that φ is locally injective.

▶ **Claim 9.** *If (G, H, φ) is in the normal form then $p(G', H', \varphi') \leq p(G, H, \varphi)$. If additionally φ is not locally injective after suppressing in G all degree-2 vertices incident to an edge induced by a cluster, the inequality is strict; that is, $p(G', H', \varphi') < p(G, H, \varphi)$.*

Furthermore, if G is connected and every connected component C induced by V_ν, for all $\nu \in V(H)$, has pipe-degree at most 2, then $|E_p(G')| \leq |E_p(G)|$.

Proof. We prove the first part of the claim and along the way establish the second part. Let ψ_0 be a $\mathbb{Z}_2$-approximation of (G, H, φ). Note that in G' the pipe edges incident to a central vertex $v_s \in V_s$ (every such v_s has degree at least 3) and edges in H' incident to $\varphi'(v_s)$ contribute together zero towards $p(G', H', \varphi')$. Let $H_0' = H' \setminus \{v_s^* |\ v_s \in V_s\}$.

($*$) The number of edges in H_0' is at least $|V(H_0')| - c = |E(H)| - c$,
where c is the number of connected components of H_0' that are trees.

We use this fact together with a simple charging scheme in terms of an injective mapping ζ defined in the next paragraph to prove the claim.

Suppose for a while that H_0' is connected. The set of pipe edges of G' not incident to any $v_s \in V_s$ forms a matching M' in G' by Definitions 4 and 5. Let $D(v)$, $v \in V(G')$, be the

connected component of $G' \setminus E_p(G')$ containing the vertex v. By the first property of the components $G[V \setminus V_s]$ in Definition 4, for every $v \in V(M')$, the component $D(v)$ contains at least one former pipe edge, i.e., a pipe edge in (G, H, φ). Let V_p be the set of vertices in G' incident to these former pipe edges.

We construct an injective mapping ζ from the set $V(M')$ to V_p. The mapping ζ maps a vertex $v \in V(M')$ to a closest vertex (in terms of the graph theoretical distance in $D(v)$) in $V_p \cap V(D(v))$. The mapping ζ is injective by the fact, that in the corresponding subdivided normal form, every connected component of $G[V_\nu]$ of pipe-degree 2, for $\nu \in V(H)$, contains at most one central vertex. Indeed, recall that this central vertex is suppressed in the normal form and the edge thereby created becomes a pipe edge e in H', and thus, belongs to M'. Each end vertex v of e is mapped by ζ to a vertex u such that $\varphi(u) = \varphi(v)$. Thus, the injectivity could be violated only by the end vertices of e. However, this cannot happen since every connected component of $G[V_\nu]$ is a tree. The injectivity of ζ implies that $2|M'|$ is upper bounded by $2|E_p(G)|$, and therefore $|M'|$ is upper bounded by $|E_p(G)|$, which proves the second part of the claim. Furthermore, $|E_p(G)| = |M'|$ only if after suppressing all vertices of degree 2 incident to an edge induced by a cluster, φ is locally injective, and H'_0 contains a cycle.

If H'_0 is disconnected, then we have $|M'| \leq |E_p(G)| - c$, where the inequality is strict if φ is not locally injective after suppressing all degree-2 vertices incident to an edge induced by a cluster. Indeed, if $|M'| = |E_p(G)| - c$, then there exist exactly $2c$ vertices v, $v \in e \in E_p(G)$, that are not in the image of the map ζ. However, there are at least $2c$ vertices in G' each of which is mapped by φ' to a vertex of degree at most 1 in H'_0. This follows since a connected component of H'_0, that is an isolated vertex ν, contributes at least two end vertices of an edge $e \in E_p(G)$ such that $\varphi'(e) = \nu$; and a connected component of H'_0, that is a tree, has at least two leaves each of which contributes by at least one end vertex of an edge in $E_p(G)$ mapped to it by φ'. Hence, if $|M'| = |E_p(G)| - c$ then all the vertices that are not contained in the image of ζ, are accounted for by these $2c$ vertices.

Putting it together, we have $|M'| \leq |E_p(G)| - c$ and $(*)$ $|E(H)| - c \leq |E(H'_0)|$, where the first inequality is strict if φ is not locally injective after suppressing all degree-2 vertices incident to an edge induced by a cluster. Since the remaining pipe edges of G' and edges of H' contribute together zero towards $p(G', H', \varphi')$, summing up the inequalities concludes the proof. ◀

▶ **Claim 10.** *Suppose that φ is locally injective after suppressing all degree-2 vertices incident to an edge induced by a cluster. Applying the derivative $|E_p(G)|$ many times yields an instance in which no connected component of G is a path.*

Proof of Theorem 1. We assume that every edge of H is in the image of φ and proceed by induction on $p(G, H, \varphi)$. First, we discuss the inductive step. By Claim 8, we assume that (G, H, φ) is in the normal form, which leaves $p(G, H, \varphi)$ unchanged. Suppose that φ is not locally injective after suppressing degree-2 vertices incident to an edge induced by a cluster. Derivating (G, H, φ) decreases $p(G, H, \varphi)$ by Claim 9. By Claims 6 and 7, $(G', H', \varphi', \psi'_0)$ is a clone of (G, H, φ, ψ_0), where ψ'_0 is obtained by Claim 6. Hence, in this case we are done by induction. Thus, we assume that φ is locally injective, which includes the case when $p(G, H, \varphi) = 0$. This means that either we reduced G to an empty graph, or every connected component C of $G[V_\nu]$, for every $\nu \in V(H)$, is a single vertex. We suppose that G is not a trivial graph, since otherwise we are done. The proof will split into two cases, the **acyclic** and the **cyclic** case below, but first we introduce some tools from [13] that we use extensively in the argument.

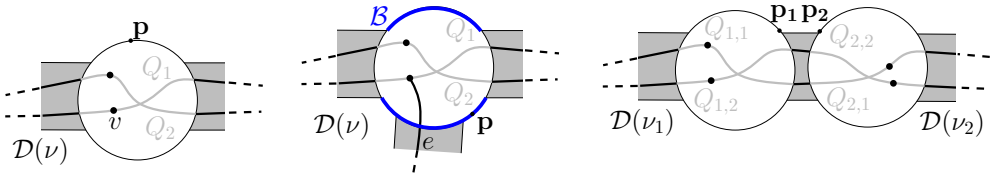


Figure 4 A pair of ν-diagonals Q_1 and Q_2, and a vertex v such that $v <_{\mathbf{p}} Q_1$ and $Q_2 <_{\mathbf{p}} Q_1$ (left). The edge e forcing $Q_1 <_{\mathbf{p}} Q_2$ (middle). The relation $Q_{1,1} <_{\mathbf{p_1}} Q_{2,1}$ forces $Q_{1,2} <_{\mathbf{p_2}} Q_{2,2}$ (right).

We will work with a $\mathbb{Z}_2$-approximation ψ_0 of (G, H, φ) unless specified otherwise. Let P be a path of length 2 in G. Let the internal vertex u of P belong to $G[V_\nu]$, for some $\nu \in V(H)$. The curve obtained by intersecting the disc $\mathcal{D}(\nu)$ with P is a *ν-diagonal supported* by u. By a slight abuse of notation we denote in different drawings ν-diagonals with the same supporting vertex and joining the same pair of valves by the same symbol. Let Q be a ν-diagonal supported by a vertex u of $G[V_\nu]$. Since φ is locally injective, Q must connect a pair of distinct valves. Let $\mathbf{p}$ be a point on the boundary of the disc $\mathcal{D}(\nu)$ of ν such that $\mathbf{p}$ is not contained in any valve.

▶ **Definition 11.** (See Figure 4 (left) for an illustration.) For a vertex $v \in V_\nu$, $v \neq u$, we define $v <_{\mathbf{p}} Q$ if in the two-coloring of the complement of Q (such that connected regions sharing a non-trivial part of the boundary receive different colors) in the disc $\mathcal{D}(\nu)$, v receives the same color as the component having $\mathbf{p}$ on the boundary. Let Q_1 and Q_2 be a pair of ν-diagonals connecting the same pair of valves. We define $Q_1 <_{\mathbf{p}} Q_2$ if for the vertex v supporting Q_1 we have $v <_{\mathbf{p}} Q_2$. Analogously we define the relation $>_{\mathbf{p}}$.

Recall that H contains no multiple edges, since we do not introduce them by derivating. Thus, the upcoming Claims 12 and 13 follow easily by the same argument as (1) and (2) in [13, Theorem 13].

▶ **Claim 12.** *The relation $<_{\mathbf{p}}$ is anti-symmetric: If for a pair of ν-diagonals Q_1, Q_2 of $G[V_\nu]$ we have $Q_1 <_{\mathbf{p}} Q_2$ then $Q_1 \not>_{\mathbf{p}} Q_2$.*

By Claim 12, the relation $<_{\mathbf{p}}$ defines a tournament, that is, a complete oriented graph, on ν-diagonals joining the same pair of valves. A pair of a ν_1-diagonal Q_1 and a ν_2-diagonal Q_2 of G is *neighboring* if Q_1 and Q_2 have endpoints on the same (pipe) edge.

Let $Q_{1,i}$ and $Q_{2,i}$ be a neighboring pair of a ν_1-diagonal and a ν_2-diagonal sharing a pipe edge e_i, for $i = 1, 2$, such that $\varphi(e_1) = \varphi(e_2) = \rho = \nu_1\nu_2$. Let $\mathbf{p_1}$ and $\mathbf{p_2}$ be on the boundary of $\mathcal{D}(\nu_1)$ and $\mathcal{D}(\nu_2)$, respectively, very close to the same side of the pipe of ρ.

▶ **Claim 13.** *If $Q_{1,1} <_{\mathbf{p_1}} Q_{1,2}$ then $Q_{2,1} <_{\mathbf{p_2}} Q_{2,2}$; see Figure 4 (right) for an illustration.*

Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be a set of ν_1-diagonals and ν_2-diagonals, respectively, in G of the same cardinality such that every ν_1-diagonal in $\mathcal{D}_1$ ends on the valve of ρ and forms a neighboring pair with a ν_2-diagonal in $\mathcal{D}_2$. We require that all the diagonals in $\mathcal{D}_1$ join the same pair of valves. Let $\overrightarrow{G(\mathcal{D}_i)}$, for $i = 1, 2$, be the tournament with vertex set $\mathcal{D}_i$ defined by the relation $<_{\mathbf{p_i}}$. An oriented graph $\overrightarrow{D}$ is *strongly connected* if there exists a directed path in $\overrightarrow{D}$ from u to v for every ordered pair of vertices u and v in $V(\overrightarrow{D})$. The following claim follows from Claim 13.

▶ **Claim 14.** *If $\overrightarrow{G(\mathcal{D}_1)}$ is strongly connected then all the diagonals in $\overrightarrow{G(\mathcal{D}_2)}$ join the same pair of valves, and the oriented graph $\overrightarrow{G(\mathcal{D}_2)}$ is strongly connected.*

The part of G inside $\mathcal{D}(\nu)$ is the union of all ν-diagonals. The part of G inside $\mathcal{D}(\nu)$ is *embedded* if ψ_0 does not contain any edge crossing in $\mathcal{D}(\nu)$.

Acyclic case. In this case, we assume that for every $\nu \in V(H)$ and every $\mathbf{p} \in \partial\mathcal{D}(\nu)$ not contained in any valve, the relation $<_{\mathbf{p}}$ induces an acyclic tournament on every set of pairwise vertex-disjoint ν-diagonals joining the same pair of valves.

We show that we can embed the part of G inside every disc $\mathcal{D}(\nu)$ while respecting the relation $<_{\mathbf{p}}$ defined according to ψ_0. In other words, in every cluster we embed connected components (now just vertices) induced by V_ν together with parts of their incident pipe edges ending on valves so that the relations $Q_1 <_{\mathbf{p}} Q_2$ are preserved for every pair of ν-diagonals Q_1 and Q_2 joining the same pair of valves. Then by reconnecting the parts G inside $\mathcal{D}(\nu)$'s we obtain a required embedding of G which will conclude the proof.

By an easy application of the unified Hanani–Tutte theorem we obtain an embedding of the part of G inside $\mathcal{D}(\nu)$. We apply the theorem to an independently even drawing of an auxiliary graph $G_{aux}(\nu)$ in $\mathcal{D}(\nu)$, where the drawing is obtained as the union of the part of G inside $\mathcal{D}(\nu)$ and $\partial\mathcal{D}(\nu)$. By subdividing edges in $G_{aux}(\nu)$ we achieve that the vertices drawn in $\partial\mathcal{D}(\nu)$ are even and therefore we indeed obtain an embedding of the part of G inside $\mathcal{D}(\nu)$ as required. Suppose that in the embedding of the part of G inside $\mathcal{D}(\nu)$ we have $Q_1 >_{\mathbf{p}} Q_2$ while in the drawing ψ_0 we have $Q_1 <_{\mathbf{p}} Q_2$. For the sake of contradiction we consider the embedding with the smallest number of such pairs, and consider such pair Q_1 and Q_2 whose end points are closest to each other along the valve that contains them.

First, we assume that both Q_1 and Q_2 pass through a connected component (a single vertex) of $G[V_\nu]$ of pipe degree 2. The endpoints of Q_1 and Q_2 are consecutive along valves, since $<_{\mathbf{p}}$ is acyclic. Thus, we just exchange them thereby contradicting the choice of the embedding. Second, we show that if Q_1 passes through a connected component C_1 of $G[V_\nu]$ of pipe degree at least 3 and Q_2 passes through a component C_2 of pipe degree 2, then the relation $Q_1 >_{\mathbf{p}} Q_2$ in the drawing of ψ_0 leads to contradiction as well. Let ρ be an edge of H such that there exists an edge incident to C_1 mapped to ρ by φ and there does not exist such an edge incident to C_2, see Figure 4 (middle) for an illustration. Let $\mathcal{B}$ be the complement of the union of the valves containing the endpoints of Q_1 or Q_2 in the boundary of $\mathcal{D}(\nu)$. Suppose that the valve of ρ and $\mathbf{p}$ are contained in the same connected component of $\mathcal{B}$. It must be that $Q_1 <_{\mathbf{p}} Q_2$ in every $\mathbb{Z}_2$-approximation of (G, H, φ). If the valve of ρ and $\mathbf{p}$ are contained in the different connected components of $\mathcal{B}$, it must be that $Q_1 >_{\mathbf{p}} Q_2$ in every $\mathbb{Z}_2$-approximation of (G, H, φ), in particular also in an approximation.

Finally, we assume that Q_1 and Q_2 pass through a connected component C_1 and C_2, respectively, of $G[V_\nu]$ of pipe degree at least 3. Similarly as above, let ρ_1 and ρ_2 be edges of H such that there exists an edge incident to C_1 and C_2, respectively, mapped to ρ_1 and ρ_2 by φ, and neither Q_1 nor Q_2 ends on its valve. By applying the unified Hanani–Tutte theorem to $G_{aux}(\nu)$ as above, we have $\rho_1 \neq \rho_2$. By the same token, the valve of ρ_1 and ρ_2 are not contained in the same connected component $\mathcal{B}$. Thus, by the same argument as in the previous case it must be that either in every $\mathbb{Z}_2$-approximation of (G, H, φ) we have $Q_1 <_{\mathbf{p}} Q_2$ or in every $\mathbb{Z}_2$-approximation of (G, H, φ) we have $Q_1 >_{\mathbf{p}} Q_2$.

In order to finish the proof in the acyclic case, we would like to reconnect neighboring pairs of diagonals by curves inside the pipes without creating a crossing. Let $Q_{1,i}$ and $Q_{2,i}$, for $i = 1, 2$, be a pair of a neighboring ν_1-diagonal and ν_2-diagonal sharing a pipe edge e_i, for $i = 1, 2$, such that $\varphi(e_1) = \varphi(e_2) = \rho$. We would like the endpoints of $Q_{1,1}$ and $Q_{1,2}$ to be ordered along the valve of ρ consistently with the endpoints of $Q_{2,1}$ and $Q_{2,2}$ along the other valve of ρ. We are done if Claim 13 applies to $Q_{i,j}$'s. However, this does not have to be the case if, let's say $Q_{1,1}$ and $Q_{2,1}$, does not join the same pair of valves. Nevertheless,

by treating all the valves distinct from the valve of ρ at $\mathcal{D}(\nu_1)$ as a single valve, we see that both Claim 12 and Claim 13, in fact, apply to $<_{\mathbf{p_1}}$ and $<_{\mathbf{p_2}}$.

Cyclic case. In this case, we assume that for a vertex $\nu \in V(H)$ and $\mathbf{p} \in \partial\mathcal{D}(\nu)$ not contained in any valve, the relation $<_{\mathbf{p}}$ induces an acyclic tournament on a set of pairwise vertex-disjoint ν-diagonals joining the same pair of valves.

We consider at least three ν-diagonals $Q_1, \ldots, Q_l$ inducing a strongly connected component in the tournament defined by $<_{\mathbf{p}}$. Let $\mathbf{p_k}$ and $\mathbf{q_k}$ be endpoints of Q_k, for $k = 1, \ldots, l$. We assume that $\mathbf{p_k}$, for $k = 1, \ldots, l$, are contained in the same valve, and therefore the same holds for $\mathbf{q_k}$. By Claim 10, we assume that no connected component in G is a path. Hence, by Claim 14 every Q_k is contained in a (drawing of a) connected component of G which is a cycle. Indeed, a vertex of degree at least 3 in a connected component of G, whose vertex supports Q_k, would inevitably lead to a pair of independent edges crossing oddly in ψ_0, since we assume that φ is locally injective. Thus, by a simple inductive argument using Claim 14 and the fact that no two distinct strongly connected components in an oriented graph share a vertex we obtain the following property of $Q_1, \ldots, Q_l$.

Every endpoint $\mathbf{p_k}$ is joined by a curve in the closure of $\psi_0(G) \setminus \bigcup_{l'=1}^{l} Q_{l'}$ with an endpoint $\mathbf{q_{k'}}$. This defines a permutation π of the set $\{Q_1, \ldots, Q_l\}$, where $\pi(Q_k) = Q_{k'}$. On the one hand, each orbit in the permutation π must obviously consist of ν-diagonals supported by vertices in the same connected component of G, which is a cycle as we discussed in the previous paragraph. On the other hand, every pair of diagonals belonging to different orbits is supported by vertices in different cycles in G. Hence, the orbits of π are in a one-to-one correspondence with a subset of connected components in G all of which are cycles. Let $C_1 \ldots C_o$, $o \leq l$, denote such cycles. By a simple inductive argument which uses Claim 14, we have that every $\varphi(C_k) = W_k, \ldots, W_k$ with W_k being repeated o_k-times, where W_k is a closed walk of H and o_k is the size of the orbit corresponding to C_k. By the hypothesis of Theorem 1 we assume that $(C_k, H|_{\varphi(C_k)}, \varphi|_{C_k})$, for $k = 1, \ldots, o$, is a positive instance.

By the previous assumption, if the number of negative signs on the edges in W_k (counted with multiplicities) is even then $o_k = 1$. Indeed, a closed neighborhood of an approximation ψ_{C_k} (which is an embedding) of $(C_k, H|_{\varphi(C_k)}, \varphi|_{C_k})$ is the annulus, in which (the image of) ψ_{C_k} is a non-self intersecting closed piecewise linear curve. Analogously, we show that if the number of negative signs on the edges in W_k (counted with multiplicities) is odd then $o_k \leq 2$, and $o_k = 1$ for at most a single value of k, i.e., if $o_{k_1} = o_{k_2} = 1$ then $k_1 = k_2$. Suppose that the previous claim holds for every $k = 1, \ldots, o$. Since $l \geq 3$ and $o_k \leq 2$ for $k = 1, \ldots, o$, we have that $o \geq 2$. We assume that $o_2 \leq o_1$. We remove the cycle C_1 from G and apply induction. Let ψ be an approximation of $(G \setminus C_1, H, \varphi|_{G \setminus C_1})$ that we obtain by the induction hypothesis. We construct the desired approximation of (G, H, φ) by extending ψ to G as follows. We embed C_1 alongside C_2 while satisfying (A) and (B) for (G, H, φ), which is possible since $1 \leq o_2 \leq o_1 \leq 2$.

It remains to show the claim. For the sake of contradiction we assume that $o_{k_1} = o_{k_2} = 1$, for $k_1 \neq k_2$. The curves $\psi_0(C_{k_1})$ and $\psi_0(C_{k_2})$ are one-sided and homotopic, and therefore they must cross an odd number of times in $\psi_0(G)$ (contradiction). Finally, for the sake of contradiction suppose that for some k, we have $o_k \geq 3$ and that there exists an approximation ψ_{C_k} of $(C_k, H|_{\varphi(C_k)}, \varphi|_{C_k})$ (which is an embedding). If o_k is odd we replace (the image of) ψ_{C_k} by the boundary of its small closed neighborhood, which is connected. Thus, we can and shall assume that o_k is even and still bigger than 2. A closed neighborhood of ψ_{C_k} is the Möbius band. By lifting $\psi(C_k)$ to the annulus via the double cover of the Möbius band, we obtain a piecewise linear closed non-self intersecting curve winding $o_k/2 > 1$ times around its center (contradiction). ◀

References

1 Hugo A. Akitaya, Greg Aloupis, Jeff Erickson, and Csaba Tóth. Recognizing weakly simple polygons. In *32nd International Symposium on Computational Geometry (SoCG 2016)*, volume 51 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 8:1–8:16, 2016.

2 Hugo A. Akitaya, Radoslav Fulek, and Csaba D. Tóth. Recognizing weak embeddings of graphs. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 274–292, 2018. doi:10.1137/1.9781611975031.20.

3 Patrizio Angelini, Giordano Da Lozzo, Giuseppe Di Battista, and Fabrizio Frati. Strip planarity testing for embedded planar graphs. *Algorithmica*, 77(4):1022–1059, 2017.

4 Patrizio Angelini, Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, Maurizio Patrignani, and Ignaz Rutter. *Beyond Level Planarity*, pages 482–495. Springer International Publishing, Cham, 2016.

5 Patrizio Angelini and Giordano Da Lozzo. Clustered Planarity with Pipes. In Seok-Hee Hong, editor, *27th International Symposium on Algorithms and Computation (ISAAC 2016)*, volume 64 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 13:1–13:13, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ISAAC.2016.13.

6 Therese C. Biedl. Drawing planar partitions III: Two constrained embedding problems, 1998.

7 Hsien-Chih Chang, Jeff Erickson, and Chao Xu. Detecting weakly simple polygons. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1655–1670. SIAM, Philadelphia, PA, 2015. doi:10.1137/1.9781611973730.110.

8 F. Cortese and G. Di Battista. Clustered planarity (invited lecture). In *Twenty-first annual symposium on Computational Geometry (proc. SoCG 05)*, pages 30–32, 2005.

9 Pier Francesco Cortese, Giuseppe Di Battista, Maurizio Patrignani, and Maurizio Pizzonia. On embedding a cycle in a plane graph. *Discrete Math.*, 309(7):1856–1869, 2009. doi:10.1016/j.disc.2007.12.090.

10 Qing-Wen Feng, Robert F. Cohen, and Peter Eades. How to draw a planar clustered graph. In *Computing and combinatorics (Xi'an, 1995)*, volume 959 of *Lecture Notes in Comput. Sci.*, pages 21–30. Springer, Berlin, 1995. doi:10.1007/BFb0030816.

11 Qing-Wen Feng, Robert F. Cohen, and Peter Eades. Planarity for clustered graphs. In *Algorithms — ESA '95*, volume 979 of *Lecture Notes in Comput. Sci.*, pages 213–226. Springer Berlin Heidelberg, 1995.

12 Radoslav Fulek. Embedding graphs into embedded graphs. In *28th International Symposium on Algorithms and Computation, ISAAC 2017, December 9-12, 2017, Phuket, Thailand*, pages 34:1–34:12, 2017. doi:10.4230/LIPIcs.ISAAC.2017.34.

13 Radoslav Fulek, Jan Kynčl, Igor Malinović, and Dömötör Pálvölgyi. Clustered planarity testing revisited. *Electron. J. Combin.*, 22(4):Paper 4.24, 29 pp., 2015.

14 Radoslav Fulek, Jan Kynčl, and Dömötör Pálvölgyi. Unified Hanani–Tutte theorem. *Electr. J. Comb.*, 24(3):P3.18, 2017. URL: http://www.combinatorics.org/ojs/index.php/eljc/article/view/v24i3p18.

15 Radoslav Fulek, Michael Pelsmajer, and Marcus Schaefer. Hanani–Tutte for radial planarity II. In Yifan Hu and Martin Nöllenburg, editors, *Graph Drawing and Network Visualization: 24th International Symposium, GD 2016, Athens, Greece, September 19-21, 2016, Revised Selected Papers*, pages 468–481, Cham, 2016. Springer International Publishing.

16 Radoslav Fulek, Michael Pelsmajer, and Marcus Schaefer. Hanani–Tutte for radial planarity. *Journal of Graph Algorithms and Applications*, 21(1):135–154, 2017.

17 Radoslav Fulek, Michael J. Pelsmajer, Marcus Schaefer, and Daniel Štefankovič. Hanani–Tutte, monotone drawings, and level-planarity. In *Thirty essays on geometric graph theory*, pages 263–287. Springer, New York, 2013. doi:10.1007/978-1-4614-0110-0_14.

18 François Le Gall. Powers of tensors and fast matrix multiplication. arXiv:1401.7714, 2014.

19 Carsten Gutwenger, Michael Jünger, Sebastian Leipert, Petra Mutzel, Merijam Percan, and René Weiskircher. Advances in c-planarity testing of clustered graphs. In Michael T. Goodrich and Stephen G. Kobourov, editors, *Graph Drawing: 10th International Symposium, GD 2002 Irvine, CA, USA, August 26–28, 2002 Revised Papers*, pages 220–236, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. doi:10.1007/3-540-36151-0_21.

20 Haim Hanani. Über wesentlich unplättbare Kurven im drei-dimensionalen Raume. *Fundamenta Mathematicae*, 23:135–142, 1934.

21 Seok hee Hong and Hiroshi Nagamochi. Two-page book embedding and clustered graph planarity. *Theoretical Computing Science*, 2016.

22 John Hopcroft and Robert Tarjan. Efficient planarity testing. *J. ACM*, 21(4):549–568, 1974.

23 Piotr Minc. Embedding of simplicial arcs into the plane. *Topology Proc.*, 22(Summer):305–340, 1997.

24 Bojan Mohar. A linear time algorithm for embedding graphs in an arbitrary surface. *SIAM J. Discrete Math.*, 12(1):6–26, 1999.

25 Bojan Mohar and Carsten Thomassen. *Graphs on surfaces*. Johns Hopkins University Pres, 2001.

26 Michael J. Pelsmajer, Marcus Schaefer, and Daniel Štefankovič. Removing even crossings. *J. Combin. Theory Ser. B*, 97(4):489–500, 2007.

27 Dušan Repovš and Arkadij B. Skopenkov. A deleted product criterion for approximability of maps by embeddings. *Topology Appl.*, 87(1):1–19, 1998. doi:10.1016/S0166-8641(97)00121-1.

28 Marcus Schaefer. Hanani–Tutte and related results. In *Geometry—intuitive, discrete, and convex*, volume 24 of *Bolyai Soc. Math. Stud.*, pages 259–299. János Bolyai Math. Soc., Budapest, 2013.

29 Marcus Schaefer. Toward a theory of planarity: Hanani-Tutte and planarity variants. *J. Graph Algorithms Appl.*, 17(4):367–440, 2013. doi:10.7155/jgaa.00298.

30 Mikhail Skopenkov. On approximability by embeddings of cycles in the plane. *Topology Appl.*, 134(1):1–22, 2003. doi:10.1016/S0166-8641(03)00069-5.

31 Carsten Thomassen. The graph genus problem is NP-complete. *Journal of Algorithms*, 10(4):568–576, 1989. doi:10.1016/0196-6774(89)90006-0.

32 W. T. Tutte. Toward a theory of crossing numbers. *J. Combinatorial Theory*, 8:45–53, 1970.

33 Douglas H. Wiedemann. Solving sparse linear equations over finite fields. *IEEE Trans. Inform. Theory*, 32(1):54–62, 1986. doi:10.1109/TIT.1986.1057137.

34 Virginia Vassilevska Williams. Multiplying matrices faster than Coppersmith-Winograd. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, STOC '12, pages 887–898, 2012.

The $\mathbb{Z}_2$-Genus of Kuratowski Minors

Radoslav Fulek[1]
IST Austria
Am Campus 1, 3400 Klosterneuburg, Austria
radoslav.fulek@ist.ac.at
https://orcid.org/0000-0001-8485-1774

Jan Kynčl[2]
Department of Applied Mathematics and Institute for Theoretical Computer Science, Charles University, Faculty of Mathematics and Physics
Malostranské nám. 25, 118 00 Praha 1, Czech Republic
kyncl@kam.mff.cuni.cz
https://orcid.org/0000-0003-4908-4703

Abstract

A drawing of a graph on a surface is *independently even* if every pair of nonadjacent edges in the drawing crosses an even number of times. The $\mathbb{Z}_2$*-genus* of a graph G is the minimum g such that G has an independently even drawing on the orientable surface of genus g. An unpublished result by Robertson and Seymour implies that for every t, every graph of sufficiently large genus contains as a minor a projective $t \times t$ grid or one of the following so-called t*-Kuratowski graphs*: $K_{3,t}$, or t copies of K_5 or $K_{3,3}$ sharing at most 2 common vertices. We show that the $\mathbb{Z}_2$-genus of graphs in these families is unbounded in t; in fact, equal to their genus. Together, this implies that the genus of a graph is bounded from above by a function of its $\mathbb{Z}_2$-genus, solving a problem posed by Schaefer and Štefankovič, and giving an approximate version of the Hanani–Tutte theorem on orientable surfaces.

2012 ACM Subject Classification Mathematics of computing → Graphs and surfaces

Keywords and phrases Hanani–Tutte theorem, genus of a graph, $\mathbb{Z}_2$-genus of a graph, Kuratowski graph

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.40

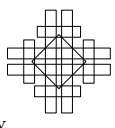
Related Version A full version of this paper is available at `https://arxiv.org/abs/1803.05085`

Acknowledgements The research was partially performed during the BIRS workshop "Geometric and Structural Graph Theory" (17w5154) in August 2017 and during a workshop on topological combinatorics organized by Arnaud de Mesmay and Xavier Goaoc in September 2017. We thank Zdeněk Dvořák, Xavier Goaoc and Pavel Paták for helpful discussions. We also thank Bojan Mohar, Paul Seymour, Gelasio Salazar, Jim Geelen and John Maharry for information about their unpublished results related to Claim 5.

1 Introduction

The *genus* $\mathrm{g}(G)$ of a graph G is the minimum g such that G has an embedding on the orientable surface M_g of genus g. We say that two edges in a graph are *independent* (also

[1] Supported by Austrian Science Fund (FWF): M2281-N35.
[2] Supported by project 16-01602Y of the Czech Science Foundation (GAČR), by the Czech-French collaboration project EMBEDS II (CZ: 7AMB17FR029, FR: 38087RM) and by Charles University project UNCE/SCI/004.




nonadjacent) if they do not share a vertex. The $\mathbb{Z}_2$-*genus* $\mathrm{g}_0(G)$ of a graph G is the minimum g such that G has a drawing on M_g with every pair of independent edges crossing an even number of times. Clearly, every graph G satisfies $\mathrm{g}_0(G) \leq \mathrm{g}(G)$.

The Hanani–Tutte theorem [13, 24] states that $\mathrm{g}_0(G) = 0$ implies $\mathrm{g}(G) = 0$. The theorem is usually stated in the following form, with the optional adjective "strong".

▶ **Theorem 1** (The (strong) Hanani–Tutte theorem [13, 24]). *A graph is planar if it can be drawn in the plane so that no pair of independent edges crosses an odd number of times.*

Theorem 1 gives an interesting algebraic characterization of planar graphs that can be used to construct a simple polynomial algorithm for planarity testing [21].

Pelsmajer, Schaefer and Stasi [17] extended the strong Hanani–Tutte theorem to the projective plane, using the list of minimal forbidden minors. Colin de Verdière et al. [7] recently provided an alternative proof, which does not rely on the list of forbidden minors.

▶ **Theorem 2** (The (strong) Hanani–Tutte theorem on the projective plane [7, 17]). *If a graph G has a drawing on the projective plane such that every pair of independent edges crosses an even number of times, then G has an embedding on the projective plane.*

Whether the strong Hanani–Tutte theorem can be extended to some other surface than the plane or the projective plane has been an open problem. Schaefer and Štefankovič [22] conjectured that $\mathrm{g}_0(G) = \mathrm{g}(G)$ for every graph G and showed that a minimal counterexample to the extension of the strong Hanani–theorem on any surface must be 2-connected. Recently, a counterexample has been found on the orientable surface of genus 4 [11].

▶ **Theorem 3** ([11]). *There is a graph G with $\mathrm{g}(G) = 5$ and $\mathrm{g}_0(G) \leq 4$. Consequently, for every positive integer k there is a graph G with $\mathrm{g}(G) = 5k$ and $\mathrm{g}_0(G) \leq 4k$.*

Schaefer and Štefankovič [22] also posed the following natural question.

▶ **Problem 1** ([22]). *Is there a function f such that $\mathrm{g}(G) \leq f(\mathrm{g}_0(G))$ for every graph G?*

We give a positive answer to Problem 1 for several families of graphs, which we conjectured to be "unavoidable" as minors in graphs of large genus. Recently we have found that a similar Ramsey-type statement is a folklore unpublished result in the graph-minors community. Together, these results would imply a positive solution to Problem 1 for all graphs. We state the results in detail in Sections 3 and 4 after giving necessary definitions in Section 2.

2 Preliminaries

2.1 Graphs on surfaces

We refer to the monograph by Mohar and Thomassen [16] for a detailed introduction into surfaces and graph embeddings. By a *surface* we mean a connected compact 2-dimensional topological manifold. Every surface is either *orientable* (has two sides) or *nonorientable* (has only one side). Every orientable surface S is obtained from the sphere by attaching $g \geq 0$ *handles*, and this number g is called the *genus* of S. Similarly, every nonorientable surface S is obtained from the sphere by attaching $g \geq 1$ *crosscaps*, and this number g is called the *(nonorientable) genus* of S. The simplest orientable surfaces are the sphere (with genus 0) and the torus (with genus 1). The simplest nonorientable surfaces are the projective plane (with genus 1) and the Klein bottle (with genus 2). We denote the orientable surface of genus g by M_g, and the nonorientable surface of genus g by N_g.

Let $G = (V, E)$ be a graph or a multigraph with no loops, and let S be a surface. A *drawing* of G on S is a representation of G where every vertex is represented by a unique point in S and every edge e joining vertices u and v is represented by a simple curve in S joining the two points that represent u and v. If it leads to no confusion, we do not distinguish between a vertex or an edge and its representation in the drawing and we use the words "vertex" and "edge" in both contexts. We assume that in a drawing no edge passes through a vertex, no two edges touch, every edge has only finitely many intersection points with other edges and no three edges cross at the same inner point. In particular, every common point of two edges is either their common endpoint or a crossing.

A drawing of G on S is an *embedding* if no two edges cross. A *face* of an embedding of G on S is a connected component of the topological space obtained from S by removing all the edges and vertices of G. A 2*-cell embedding* is an embedding whose each face is homeomorphic to an open disc. The *facewidth* (also called *representativity*) $\mathrm{fw}(\mathcal{E})$ of an embedding $\mathcal{E}$ on a surface S of positive genus is the smallest nonnegative integer k such that there is a closed noncontractible curve in S intersecting $\mathcal{E}$ in k vertices.

The *rotation* of a vertex v in a drawing of G on an orientable surface is the clockwise cyclic order of the edges incident to v. We will represent the rotation of v by the cyclic order of the other endpoints of the edges incident to v. The *rotation system* of a drawing is the set of rotations of all vertices.

The *Euler characteristic* of a surface S of genus g, denoted by $\chi(S)$, is defined as $\chi(S) = 2 - 2g$ if S is orientable, and $\chi(S) = 2 - g$ if S is nonorientable. Equivalently, if v, e and f denote the numbers of vertices, edges and faces, respectively, of a 2-cell embedding of a graph on S, then $\chi(S) = v - e + f$. The *Euler genus* $\mathrm{eg}(S)$ of S is defined as $2 - \chi(S)$. In other words, the Euler genus of S is equal to the genus of S if S is nonorientable, and to twice the genus of S if S is orientable.

An edge in a drawing is *even* if it crosses every other edge an even number of times. A drawing of a graph is *even* if all its edges are even. A drawing of a graph is *independently even* if every pair of independent edges in the drawing crosses an even number of times. In the literature, the notion of $\mathbb{Z}_2$*-embedding* is used to denote both an even drawing [5] and an independently even drawing [22].

The *genus* $\mathrm{g}(G)$ and the $\mathbb{Z}_2$*-genus* $\mathrm{g}_0(G)$ of a graph G have been defined in the introduction, as parameters related to drawings on orientable surfaces. The following two "Euler" analogues involve drawings on both orientable and nonorientable surfaces. The *Euler genus* $\mathrm{eg}(G)$ of G is the minimum g such that G has an embedding on a surface of Euler genus g. The *Euler* $\mathbb{Z}_2$*-genus* $\mathrm{eg}_0(G)$ of G is the minimum g such that G has an independently even drawing on a surface of Euler genus g.

The *embedding scheme* of a drawing $\mathcal{D}$ on a surface S consists of the rotation system and a signature $+1$ or -1 assigned to every edge, representing the parity of the number of crosscaps the edge is passing through. If S is orientable, the embedding scheme can be given just by the rotation system. The following weak analogue of the Hanani–Tutte theorem was proved by Cairns and Nikolayevsky [5] for orientable surfaces and then extended by Pelsmajer, Schaefer and Štefankovič [18] to nonorientable surfaces.

▶ **Theorem 4** (The weak Hanani–Tutte theorem on surfaces [5, Lemma 3], [18, Theorem 3.2])**.** *If a graph G has an even drawing $\mathcal{D}$ on a surface S, then G has an embedding on S that preserves the embedding scheme of $\mathcal{D}$.*

A simple closed curve γ in a surface S is 1*-sided* if it has a small neighborhood homeomorphic to the Möbius strip, and 2*-sided* if it has a small neighborhood homeomorphic to

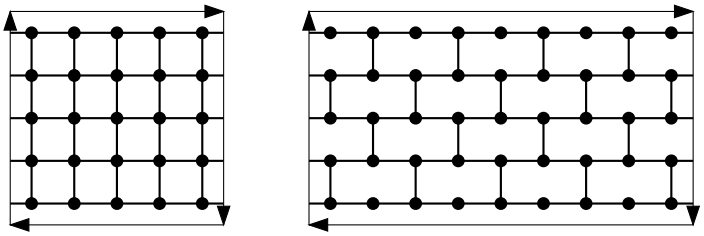


Figure 1 Left: a projective 5×5 grid. Right: a projective 5-wall.

the cylinder. We say that γ is *separating* in S if the complement $S \setminus \gamma$ has two components, and *nonseparating* if $S \setminus \gamma$ is connected. Note that on an orientable surface every simple closed curve is 2-sided, and every 1-sided simple closed curve (on a nonorientable surface) is nonseparating.

2.2 Special graphs

2.2.1 Projective grids and walls

For a positive integer n we denote the set $\{1, \ldots, n\}$ by $[n]$. Let $r, s \geq 3$. The *projective* $r \times s$ *grid* is the graph with vertex set $[r] \times [s]$ and edge set

$$\{\{(i,j),(i',j')\}; |i-i'| + |j-j'| = 1\} \ \cup \ \{\{(i,1),(r+1-i,s)\}; i \in [r]\}.$$

In other words, the projective $r \times s$ grid is obtained from the planar $r \times (s+1)$ grid by identifying pairs of opposite vertices and edges in its leftmost and rightmost column. See Figure 1, left. The projective $t \times t$ grid has an embedding on the projective plane with facewidth t. By the result of Robertson and Vitray [20], [16, p. 171], the embedding is unique if $t \geq 4$. Hence, for $t \geq 4$ the genus of the projective $t \times t$ grid is equal to $\lfloor t/2 \rfloor$ by the result of Fiedler, Huneke, Richter and Robertson [10], [16, Theorem 5.8.1].

Since grids have vertices of degree 4, it is more convenient for us to consider their subgraphs of maximum degree 3, called walls. For an odd $t \geq 3$, a *projective* t*-wall* is obtained from the projective $t \times (2t-1)$ grid by removing edges $\{(i,2j),(i+1,2j)\}$ for i odd and $1 \leq j \leq t-1$, and edges $\{(i,2j-1),(i+1,2j-1)\}$ for i even and $1 \leq j \leq t$. Similarly, for an even $t \geq 4$, a *projective* t*-wall* is obtained from the projective $t \times 2t$ grid by removing edges $\{(i,2j),(i+1,2j)\}$ for i odd and $1 \leq j \leq t$, and edges $\{(i,2j-1),(i+1,2j-1)\}$ for i even and $1 \leq j \leq t$. The projective t-wall has maximum degree 3 and can be embedded on the projective plane as a "twisted wall" with inner faces bounded by 6-cycles forming the "bricks", and with the "outer" face bounded by a $(4t-2)$-cycle for t odd and a $4t$-cycle for t even. See Figure 1, right. This embedding has facewidth t and so again, for $t \geq 4$ the projective t-wall has genus $\lfloor t/2 \rfloor$. It is easy to see that the projective 3-wall has genus 1 since it contains a subdivision of $K_{3,3}$ and embeds on the torus.

2.2.2 Kuratowski graphs

A graph is called a *t-Kuratowski* graph [23] if it is one of the following:

(a) $K_{3,t}$,
(b) a disjoint union of t copies of K_5,
(c) a disjoint union of t copies of $K_{3,3}$,
(d) a graph obtained from t copies of K_5 by identifying one vertex from each copy to a single common vertex,

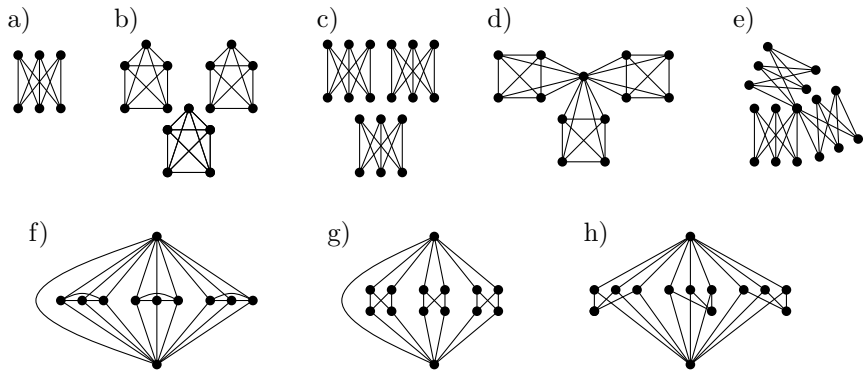


Figure 2 The eight 3-Kuratowski graphs.

(e) a graph obtained from t copies of $K_{3,3}$ by identifying one vertex from each copy to a single common vertex,
(f) a graph obtained from t copies of K_5 by identifying a pair of vertices from each copy to a common pair of vertices,
(g) a graph obtained from t copies of $K_{3,3}$ by identifying a pair of adjacent vertices from each copy to a common pair of vertices,
(h) a graph obtained from t copies of $K_{3,3}$ by identifying a pair of nonadjacent vertices from each copy to a common pair of vertices.

See Figure 2 for an illustration.

The genus of each of the t-Kuratowski graphs is known precisely. The genus of $K_{3,t}$ is $\lceil (t-2)/4 \rceil$ [4, 19], which coincides with the lower bound from Euler's formula. The genus of t copies of K_5 or $K_{3,3}$ sharing a vertex is t by the additivity of genus over blocks [1]. Finally, from a general formula by Decker, Glover and Huneke [9] it follows that the genus of t copies of K_5 or $K_{3,3}$ sharing a pair of adjacent or nonadjacent vertices is $\lceil t/2 \rceil$ if $t > 1$: cases f) and g) follow from their proof of Corollary 0.2, case h) follows from their Corollary 2.4 after realizing that $\mu(K_{3,3}) = 3$ if x, y are nonadjacent in $K_{3,3}$.

3 Ramsey-type results

The following Ramsey-type statement for graphs of large Euler genus is a folklore unpublished result.

▶ **Claim 5** (Robertson–Seymour [2, 23], unpublished). *There is a function g such that for every $t \geq 3$, every graph of Euler genus $g(t)$ contains a t-Kuratowski graph as a minor.*

For 7-connected graphs, Claim 5 follows from the result of Böhme, Kawarabayashi, Maharry and Mohar [2], stating that for every positive integer t, every sufficiently large 7-connected graph contains $K_{3,t}$ as a minor. Böhme et al. [3] later generalized this to graphs of larger connectivity and $K_{a,t}$ minors for every fixed $a > 3$.

Richter and Salazar [6] proved a similar statement for graph-like continua.

We obtain an analogous Ramsey-type statement for graphs of large genus as an almost direct consequence of Claim 5.

▶ **Theorem 6.** *Claim 5 implies that there is a function h such that for every $t \geq 3$, every graph of genus $h(t)$ contains, as a minor, a t-Kuratowski graph or the projective t-wall.*

We give a detailed proof of Theorem 6 in the full version of this paper [12].

4 Our results

As our main result we complete a proof that the $\mathbb{Z}_2$-genus of each t-Kuratowski graph and the projective t-wall grows to infinity with t; in fact, the $\mathbb{Z}_2$-genus of each of these graphs is equal to their genus. Schaefer and Štefankovič [22] proved this for those t-Kuratowski graphs that consist of t copies of K_5 or $K_{3,3}$ sharing at most one vertex. For the projective t-wall, the result follows directly from the weak Hanani–Tutte theorem on orientable surfaces [5, Lemma 3]: indeed, all vertices of the projective t-wall have degree at most 3, therefore pairs of adjacent edges crossing oddly in an independently even drawing can be redrawn in a small neighborhood of their common vertex so that they cross evenly, and the weak Hanani–Tutte theorem can be applied. Thus, the remaining cases are t-Kuratowski graphs of type a), f), g) and h).

▶ **Theorem 7.** *For every $t \geq 3$, the $\mathbb{Z}_2$-genus of each t-Kuratowski graph of type a), f), g) and h) is equal to its genus. In particular,*

(a) $\mathrm{g}_0(K_{3,t}) \geq \lceil (t-2)/4 \rceil$, *and*

(b) *if G consists of t copies of K_5 or $K_{3,3}$ sharing a pair of adjacent or nonadjacent vertices, then $\mathrm{g}_0(G) \geq \lceil t/2 \rceil$.*

Combining Theorem 7 with the result of Schaefer and Štefankovič [22] and the simple argument for the projective t-wall we obtain the following result.

▶ **Corollary 8.** *For every $t \geq 3$, the $\mathbb{Z}_2$-genus of each t-Kuratowski graph and the projective t-wall is equal to its genus.*

Combining Corollary 8 with Theorem 6 we get the following implication.

▶ **Corollary 9.** *Claim 5 implies a positive answer to Problem 1.*

5 Lower bounds on the $\mathbb{Z}_2$-genus

In this section we prove Theorem 7 for the t-Kuratowski graphs of type a), f), g) and h).

The fact that the $\mathbb{Z}_2$-genus of $K_{3,t}$ or the other t-Kuratowski graphs is unbounded when t goes to infinity is not obvious at first sight. The traditional lower bound on the genus of $K_{3,t}$ relies on Euler's formula and the notion of a face. However, there is no analogue of a "face" in an independently even drawing, and the rotations of vertices no longer "matter". We thus need different tools to compute the $\mathbb{Z}_2$-genus.

5.1 $\mathbb{Z}_2$-homology of curves

We refer to Hatcher's textbook [14] for an excellent general introduction to homology theory. Unfortunately, we were unable to find a more compact treatment of the homology theory for curves on surfaces in the literature, thus we sketch here the main aspects that are most important for us.

We will use the $\mathbb{Z}_2$-homology of closed curves on surfaces. That is, for a given surface S, we are interested in its first homology group with coefficients in $\mathbb{Z}_2$, denoted by $H_1(S;\mathbb{Z}_2)$. It is well-known that for each $g \geq 0$, the first homology group $H_1(M_g;\mathbb{Z}_2)$ of M_g is isomorphic to $\mathbb{Z}_2^{2g}$ [14, Example 2A.2. and Corollary 3A.6.(b)]. This fact was crucial in establishing the weak Hanani–Tutte theorem on M_g [5, Lemma 3].

To every closed curve γ in M_g one can assign its homology class $[\gamma] \in H_1(S;\mathbb{Z}_2)$, and this assignment is invariant under continuous deformation (homotopy). In particular, the

homology class of each contractible curve is 0. More generally, the homology class of each separating curve in M_g is 0 as well. Moreover, if γ is obtained by a composition of γ_1 and γ_2, the homology classes satisfy $[\gamma] = [\gamma_1] + [\gamma_2]$. The assignment of homology classes to closed curves is naturally extended to formal integer combinations of the closed curves, called *cycles*, and so $[\gamma]$ can be considered as a set of cycles. Since we are interested in homology with coefficients in $\mathbb{Z}_2$, it is sufficient to consider cycles with coefficients in $\mathbb{Z}_2$, which may also be regarded as finite sets of closed curves.

If γ_1 and γ_2 are cycles in M_g that cross in finitely many points and have no other points in common, we denote by $\mathrm{cr}(\gamma_1, \gamma_2)$ the number of their common crossings. We use the following well-known fact, which may be seen as a consequence of the Jordan curve theorem.

▶ **Fact 10.** *Let $\gamma_1' \in [\gamma_1]$ and $\gamma_2' \in [\gamma_2]$ be a pair of cycles in M_g such that the intersection number $\mathrm{cr}(\gamma_1', \gamma_2')$ is defined and is finite. Then*

$$\mathrm{cr}(\gamma_1', \gamma_2') \equiv \mathrm{cr}(\gamma_1, \gamma_2) \pmod 2.$$

Fact 10 allows us to define a group homomorphism (which is also a bilinear form)

$$\Omega_{M_g} : H_1(M_g; \mathbb{Z}_2) \times H_1(M_g; \mathbb{Z}_2) \to \mathbb{Z}_2$$

such that

$$\Omega_{M_g}([\gamma_1], [\gamma_2]) = \mathrm{cr}(\gamma_1, \gamma_2) \bmod 2$$

whenever $\mathrm{cr}(\gamma_1, \gamma_2)$ is defined and is finite. Cairns and Nikolayevsky [5] call Ω_{M_g} the *intersection form* on M_g. Clearly, Ω_{M_g} is symmetric and $\Omega_{M_g}([\gamma], [\gamma]) = 0$ for every cycle γ, since simple closed curves in M_g are 2-sided, and every closed curve with finitely many self-intersections is a composition of finitely many simple closed curves.

We have the following simple observation about intersections of disjoint cycles in independently even drawings.

▶ **Observation 11** ([22, Lemma 1]). *Let $\mathcal{D}$ be an independently even drawing of a graph G on M_g. Let C_1 and C_2 be vertex-disjoint cycles in G, and let γ_1 and γ_2 be the closed curves representing C_1 and C_2, respectively, in $\mathcal{D}$. Then $\mathrm{cr}(\gamma_1, \gamma_2) \equiv 0 \pmod 2$, which implies that $\Omega_{M_g}([\gamma_1], [\gamma_2]) = 0$.*

5.2 Combinatorial representation of the $\mathbb{Z}_2$-homology of drawings

Schaefer and Štefankovič [22] used the following combinatorial representation of drawings of graphs on M_g. First, every drawing of a graph on M_g can be considered as a drawing on the nonorientable surface N_{2g+1}, since M_g minus a point is homeomorphic to an open subset of N_{2g+1}. The surface N_{2g+1} minus a point can be represented combinatorially as the plane with $2g+1$ *crosscaps*. A crosscap at a point x is a combinatorial representation of a Möbius strip whose boundary is identified with the boundary of a small circular hole centered in x. Informally, the main "objective" of a crosscap is to allow a set of curves intersect transversally at x without counting it as a crossing.

Every closed curve γ drawn in the plane with $2g+1$ crosscaps is assigned a vector $y_\gamma \in \{0,1\}^{2g+1}$ such that $(y_\gamma)_i = 1$ if and only if γ passes an odd number of times through the ith crosscap. When γ comes from a drawing on M_g, then y_γ has an even number of coordinates equal to 1. The vectors y_γ represent the elements of the homology group $H_1(M_g; \mathbb{Z}_2)$, and the value of the intersection form $\Omega_{M_g}([\gamma], [\gamma'])$ is equal to the scalar product

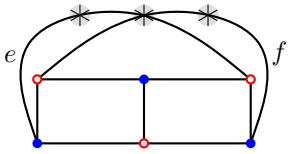


Figure 3 An embedding of $K_{3,3}$ on the torus represented as a drawing in the plane with three crosscaps. The nonzero vectors assigned to the edges are $y_e = (1, 1, 0)$ and $y_f = (0, 1, 1)$.

$y_\gamma^\top y_{\gamma'}$ over $\mathbb{Z}_2$. Analogously, we assign a vector y_e to every curve e representing an edge in a drawing of a graph in this model. See Figure 3.

We use the following two lemmata by Schaefer and Štefankovič [22].

▶ **Lemma 12** ([22, Lemma 5]). *Let G be a graph that has an independently even drawing $\mathcal{D}$ on M_g and let F be a forest in G. Then G has a drawing $\mathcal{E}$ in the plane with $2g+1$ crosscaps such that*

(1) *every pair of independent edges has an even number of common crossings outside the crosscaps, and*

(2) *every edge f of F passes through each crosscap an even number of times; that is, $y_f = 0$.*

Moreover, $\mathcal{E}$ can be obtained from $\mathcal{D}$ by a sequence of continuous deformations of edges and neighborhoods of vertices, so the homology classes of all cycles are preserved between the two drawings.

▶ **Lemma 13** ([22, Lemma 3]). *Let G be a graph that has a drawing in the plane with $2g+1$ crosscaps with every pair of independent edges having an even number of common crossings outside the crosscaps. Let d be the dimension of the vector space generated by the set $\{y_e; e \in E(G)\}$. Then G has an independently even drawing on $M_{\lfloor d/2 \rfloor}$.*

Lemma 12 and Lemma 13 imply the following corollary generalizing the strong Hanani–Tutte theorem. The proof appears in the full version of this paper.

▶ **Corollary 14.** *Let G be a connected graph with an independently even drawing on M_g such that each cycle in the drawing is homologically zero (that is, the homology class of the corresponding closed curve is 0). Then G is planar.*

Corollary 14 can be further strengthened using Lemma 12 as follows.

▶ **Lemma 15.** *Let G be a connected graph with an independently even drawing $\mathcal{D}$ on M_g. Let F be a spanning tree of G. If G is nonplanar, then there are independent edges $e, f \in E(G) \setminus E(F)$ such that the closed curves γ_e and γ_f representing the fundamental cycles of e and f, respectively, satisfy $\Omega_{M_g}([\gamma_e], [\gamma_f]) = 1$.*

Proof. Let $\mathcal{E}$ be a drawing of G from Lemma 12. By the strong Hanani–Tutte theorem, there are two independent edges e and f in G that cross an odd number of times in $\mathcal{E}$. Moreover, conditions 1) and 2) of Lemma 12 imply that none of the edges e and f is in F and so e and f cross an odd number of times in the crosscaps. This means that $y_e^\top y_f = 1$, which is equivalent to $\Omega_{M_g}([\gamma_e], [\gamma_f]) = 1$. ◀

5.3 Proof of Theorem 7a)

We will show three lower bounds on $g_0(K_{3,t})$, in the order of increasing strength and complexity of their proof.

We will adopt the following notation for the vertices of $K_{3,t}$. The vertices of degree t forming one part of the bipartition are denoted by a, b, c, and the remaining vertices by $u_0, u_1, \ldots, u_{t-1}$. Let $U = \{u_0, u_1, \ldots, u_{t-1}\}$. For each $i \in [t-1]$, let C_i be the cycle au_ibu_0 and C'_i the cycle au_icu_0.

The first lower bound, $\mathrm{g}_0(K_{3,t}) \geq \Omega(\log\log\log t)$, follows from Ramsey's theorem and the weak Hanani–Tutte theorem on surfaces. The proof appears in the full version of this paper.

The second lower bound is based on the pigeonhole principle and Corollary 14 from the previous subsection.

▶ **Proposition 16.** *We have* $\mathrm{g}_0(K_{3,t}) \geq \Omega(\log t)$.

Proof. Let $\mathcal{D}$ be an independently even drawing of $K_{3,t}$ on M_g. By the pigeonhole principle, there is a subset $I_b \subseteq [t-1]$ of size at least $(t-1)/2^{2g}$ such that all the cycles C_i with $i \in I_b$ have the same homology class in $\mathcal{D}$. Analogously, there is a subset $I_c \subseteq I_b$ of size at least $|I_b|/2^{2g}$ such that all the cycles C'_i with $i \in I_c$ have the same homology class in $\mathcal{D}$. Suppose that $t \geq 2 \cdot 16^g + 2$. Then $|I_b| \geq 2 \cdot 4^g + 1$ and $|I_c| \geq 3$. Let $i, j, k \in I_c$ be three distinct integers. We now consider the subgraph H of $K_{3,t}$ induced by the vertices a, b, c, u_i, u_j, u_k, isomorphic to $K_{3,3}$, and show that all its cycles are homologically zero. Indeed, the cycle space of H is generated by the four cycles au_ibu_j, au_ibu_k, au_icu_j and au_icu_k, and each of them is the sum (mod 2) of two cycles of the same homology class: $au_ibu_j = C_i + C_j$, $au_ibu_k = C_i + C_k$, $au_icu_j = C'_i + C'_j$ and $au_icu_k = C'_i + C'_k$. Corollary 14 now implies that H is planar, but this is a contradiction. Therefore $t \leq 2 \cdot 16^g + 1$. ◀

To prove the lower bound in Theorem 7a), we use the same general idea as in the previous proof. However, we will need a more precise lemma about drawings of $K_{3,3}$, strengthening Corollary 14 and Lemma 15. We also replace the pigeonhole principle with a linear-algebraic trick.

▶ **Lemma 17.** *Let $\mathcal{D}$ be an independently even drawing of $K_{3,3}$ on M_g. For $i \in \{1, 2\}$, let γ_i and γ'_i be the closed curves representing the cycles C_i and C'_i, respectively, in $\mathcal{D}$. The intersection numbers of their homology classes satisfy*

$$\Omega_{M_g}([\gamma_1], [\gamma'_2]) + \Omega_{M_g}([\gamma'_1], [\gamma_2]) = 1.$$

Lemma 17 is a consequence of Corollary 23. In the full version of this paper we include a direct proof using a different method.

▶ **Proposition 18.** *We have* $\mathrm{g}_0(K_{3,t}) \geq \lceil (t-2)/4 \rceil$.

Proof. Let $\mathcal{D}$ be an independently even drawing of $K_{3,t}$ on M_g. For every $i \in [t-1]$, let γ_i and γ'_i be the closed curves representing the cycles C_i and C'_i, respectively, in $\mathcal{D}$. For every $i, j \in [t-1]$, $i < j$, we apply Lemma 17 to the drawing of $K_{3,3}$ induced by the vertices a, b, c, u_0, u_i, u_j in $\mathcal{D}$. Let A be the $(t-1) \times (t-1)$ matrix with entries

$$A_{i,j} = \Omega_{M_g}([\gamma_i], [\gamma'_j]).$$

Lemma 17 implies that $A_{i,j} + A_{j,i} = 1$ whenever $i \neq j$; in other words, A is a *tournament matrix* [8]. Repeating the argument by de Caen [8], it follows that $A + A^\top$, with the addition mod 2, is the matrix with zeros on the diagonal and 1-entries elsewhere. This implies that the rank of A over $\mathbb{Z}_2$ is at least $(t-2)/2$. Hence, the rank of Ω_{M_g} is at least $(t-2)/2$, which implies $2g \geq (t-2)/2$. ◀

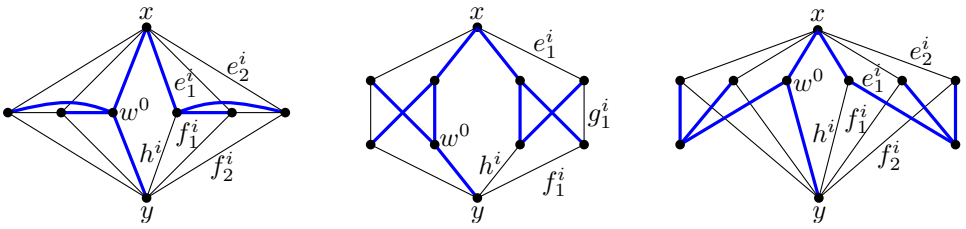


Figure 4 2-amalgamations of two Kuratowski xy-wings. The spanning tree T is drawn bold.

5.4 Proof of Theorem 7b)

Before proving Theorem 7b) we first show an asymptotic $\Omega(\log t)$ lower bound on the $\mathbb{Z}_2$-genus for a more general class of graphs that includes the t-Kuratowski graphs of types f), g) and h).

Let H be a 2-connected graph and let x, y be two nonadjacent vertices of H. Let t be a positive integer. The 2-*amalgamation* of t copies of H (with respect to x and y), denoted by $\amalg_{x,y}tH$, is the graph obtained from t disjoint copies of H by gluing all t copies of x into a single vertex and gluing all t copies of y into a single vertex. The two vertices obtained by gluing are again denoted by x and y.

An *xy-wing* is a 2-connected graph H with two nonadjacent vertices x and y such that the subgraph $H - x - y$ is connected, and the graph obtained from H by adding the edge xy is nonplanar. Clearly, the graphs $K_5 - e$ and $K_{3,3} - e$, where $e = xy$, are xy-wings, and similarly $K_{3,3}$, with nonadjacent vertices x and y, is an xy-wing. The t-Kuratowski graphs of types f) and g) are obtained from $\amalg_{x,y}t(K_5 - e)$ and $\amalg_{x,y}t(K_{3,3} - e)$, respectively, by adding the edge xy, whereas the t-Kuratowski graph of type h) is exactly the 2-amalgamation $\amalg_{x,y}t(K_{3,3})$. See Figure 4 for an illustration of 2-amalgamations of two xy-wings.

Let H be an xy-wing. We will use the following notation. Let w be a vertex of H adjacent to y and let F' be a spanning tree of $H - x - y$. Let F be a spanning tree of $H - y$ extending F'. In the 2-amalgamation $\amalg_{x,y}tH$ we distinguish the ith copy of H, its vertices, edges, and subgraphs, by the superscript $i \in \{0, 1, \ldots, t-1\}$. In particular, for every $i \in \{0, 1, \ldots, t-1\}$, H^i is an induced subgraph of $\amalg_{x,y}tH$, F^i is a spanning tree of $H^i - y$ and x is a leaf of F^i. For a given t, let

$$T = yw^0 + \bigcup_{i=0}^{t-1} F_i$$

be a spanning tree of $\amalg_{x,y}tH$. For every edge $e \in E(\amalg_{x,y}tH) \setminus E(T)$, let C_e be the fundamental cycle of e with respect to T; that is, the unique cycle in $T + e$.

Enumerate the edges of $E(H) \setminus E(F)$ incident to x as $e_1, \ldots, e_k$, the edges of $E(H) \setminus E(F) \setminus \{yw\}$ incident to y as $f_1, \ldots, f_l$, and the edges of $E(H - x - y) \setminus E(F)$ as $g_1, \ldots, g_m$. Let h be the edge yw. Thus, for every $i \in [t-1]$, we have $E(H^i) \setminus E(T) = \{e_1^i, \ldots, e_k^i\} \cup \{f_1^i, \ldots, f_l^i\} \cup \{g_1^i, \ldots, g_m^i\} \cup \{h^i\}$.

If C and C' are cycles in $\amalg_{x,y}tH$, we denote by $C + C'$ the element of the cycle space of $\amalg_{x,y}tH$ obtained by adding C and C' mod 2. We also regard $C + C'$ as a subgraph of $\amalg_{x,y}tH$ with no isolated vertices. Note that if C and C' are fundamental cycles sharing at least one edge then $C + C'$ is again a cycle.

▶ **Observation 19.** *Let $i \in [t-1]$.*

(a) *For every $j \in [k]$, the cycle $C_{e_j^i}$ is a subgraph of $H^i - y$.*

(b) *For every $j \in [l]$, the cycle $C_{f^i_j} + C_{h^i}$ is a subgraph of $H^i - x$.*
(c) *For every $j \in [m]$, the cycle $C_{g^i_j}$ is a subgraph of $H^i - x - y$.*
The cycles $C_{e^i_j}$ with $j \in [k]$, $C_{f^i_j} + C_{h^i}$ with $j \in [l]$, and $C_{g^i_j}$ with $j \in [m]$ generate the cycle space of H^i; in particular, they are the fundamental cycles of H^i with respect to the spanning tree $F^i + yw^i$. ◀

▶ **Corollary 20.** *Let $i, i' \in [t-1]$ be distinct indices. Then the following pairs of cycles are vertex-disjoint, for all possible pairs of indices j, j':*
(a) *$C_{e^i_j}$ and $C_{f^{i'}_{j'}} + C_{h^{i'}}$,*
(b) *$C_{f^i_j} + C_{h^i}$ and $C_{g^{i'}_{j'}}$,*
(c) *$C_{e^i_j}$ and $C_{g^{i'}_{j'}}$,*
(d) *$C_{g^i_j}$ and $C_{g^{i'}_{j'}}$.*

Our first lower bound on the $\mathbb{Z}_2$-genus of 2-amalgamations of xy-wings is similar to Proposition 16, and combines the pigeonhole principle and Corollary 14.

▶ **Proposition 21.** *Let H be an xy-wing. Then $\mathrm{g}_0(\amalg_{x,y} tH) \geq \Omega(\log t)$.*

Proof. Let $\mathcal{D}$ be an independently even drawing of $\amalg_{x,y} tH$ on M_g. For every $i \in [t-1]$ and $e \in E(H) \setminus E(F)$, let $\gamma(e^i)$ be the closed curve representing C_{e^i} in $\mathcal{D}$.

The homology class $[\gamma(e^i)]$ has one of 2^{2g} possible values in $H_1(M_g; \mathbb{Z}_2)$. Thus, if $t \geq 2^{2g(k+l+m+1)} + 2$, then there are distinct indices $i, i' \in [t-1]$ such that for every $e \in E(H) \setminus E(F)$ we have $[\gamma(e^i)] = [\gamma(e^{i'})]$. Combining this with Observation 11 and Corollary 20, for all possible pairs of indices j, j' we have

$$\Omega_{M_g}([\gamma(e^i_j)], [\gamma(f^i_{j'})] + [\gamma(h^i)]) = \Omega_{M_g}([\gamma(e^i_j)], [\gamma(f^{i'}_{j'})] + [\gamma(h^{i'})]) = 0, \tag{1}$$

$$\Omega_{M_g}([\gamma(f^i_j)] + [\gamma(h^i)], [\gamma(g^i_{j'})]) = \Omega_{M_g}([\gamma(f^i_j)] + [\gamma(h^i)], [\gamma(g^{i'}_{j'})]) = 0, \tag{2}$$

$$\Omega_{M_g}([\gamma(e^i_j)], [\gamma(g^i_{j'})]) = \Omega_{M_g}([\gamma(e^i_j)], [\gamma(g^{i'}_{j'})]) = 0, \tag{3}$$

$$\Omega_{M_g}([\gamma(g^i_j)], [\gamma(g^i_{j'})]) = \Omega_{M_g}([\gamma(g^i_j)], [\gamma(g^{i'}_{j'})]) = 0. \tag{4}$$

Let $H^{i,i'}$ be the union of the graph H^i with the unique xy-path $P^{i'}$ in $F^{i'} + yw^{i'}$. Since H is an xy-wing, the graph $H^{i,i'}$ is nonplanar. The graph $F^{i,i'} = F^i \cup P^{i'}$ is a spanning tree of $H^{i,i'}$, and $E(H^{i,i'}) \setminus E(F^{i,i'}) = E(H^i) \setminus E(T)$.

The fundamental cycle C'_{h^i} of h^i in $H^{i,i'}$ with respect to $F^{i,i'}$ is equal to $C_{h^i} + C_{h^{i'}}$. Since $[\gamma(h^i)] = [\gamma(h^{i'})]$, the cycle C'_{h^i} is homologically zero.

For every $j \in [k]$, the fundamental cycle of e^i_j in $H^{i,i'}$ with respect to $F^{i,i'}$ is $C_{e^i_j}$ and its homology class in $\mathcal{D}$ is $[\gamma(e^i_j)]$.

For every $j \in [l]$, the fundamental cycle of f^i_j in $H^{i,i'}$ with respect to $F^{i,i'}$ is $C_{f^i_j} + C_{h^{i'}}$ and its homology class is $[\gamma(f^i_{j'})] + [\gamma(h^{i'})] = [\gamma(f^i_{j'})] + [\gamma(h^i)]$.

For every $j \in [m]$, the fundamental cycle of g^i_j in $H^{i,i'}$ with respect to $F^{i,i'}$ is $C_{g^i_j}$ and its homology class in $\mathcal{D}$ is $[\gamma(g^i_j)]$.

By (1)–(4), for every pair of independent edges in $E(H^{i,i'}) \setminus E(F^{i,i'})$, the homology classes of their fundamental cycles are orthogonal with respect to Ω_{M_g}. This is a contradiction with Lemma 15 applied to $H^{i,i'}$ and the spanning tree $F^{i,i'}$. Therefore, $t \leq 2^{2g(k+l+m+1)} + 1$. ◀

To prove the lower bound in Theorem 7b), we follow the idea of the previous proof and again replace the pigeonhole principle with a linear-algebraic argument. We will also need a stronger variant of the Hanani–Tutte theorem and Lemma 15 for the graphs K_5 and $K_{3,3}$.

▶ **Lemma 22** (Kleitman [15]). *In every drawing of K_5 and $K_{3,3}$ in the plane the total number of pairs of independent edges crossing an odd number of times is odd.*

▶ **Corollary 23.** *Let $G = K_5$ or $G = K_{3,3}$. Let F be a forest in G. Let $\mathcal{E}$ be a drawing of G from Lemma 12. Then there are an odd number of pairs of independent edges e, f in $E(G) \setminus E(F)$ such that $y_e^\top y_f = 1$.* ◀

The following simple fact is a key ingredient in the proof of Lemma 22.

▶ **Observation 24.** *The graph obtained from each of K_5 and $K_{3,3}$ by removing an arbitrary pair of adjacent vertices is a cycle; in particular, all of its vertices have an even degree.* ◀

An xy-wing H is called a *Kuratowski xy-wing* if H is one of the graphs $K_5 - e$ where $e = xy$, $K_{3,3} - e$ where $e = xy$, or $K_{3,3}$; see Figure 4. Observation 24 implies the following important property of Kuratowski xy-wings.

▶ **Observation 25.** *Let H be a Kuratowski xy-wing and let u be a vertex adjacent to x in H. Then $H - x - u$ is a cycle; in particular, y is incident to exactly two edges in $H - x - u$.* ◀

In the following key lemma we keep using the notation for the 2-amalgamation $\amalg_{x,y} tH$ established earlier in this subsection.

▶ **Lemma 26.** *Let $t \geq 2$, let H be a Kuratowski xy-wing and let $\mathcal{D}$ be an independently even drawing of $\amalg_{x,y} tH$ on M_g. Then for every $i \in [0, t-1]$ the graph H^i has two cycles C_1^i and C_2^i such that*

- *(C_1^i is a subgraph of $H^i - x$ and C_2^i is a subgraph of $H^i - y$) or (C_2^i is a subgraph of $H^i - x - y$, and*
- *the closed curves γ_1^i and γ_2^i representing C_1^i and C_2^i, respectively, in $\mathcal{D}$ satisfy $\Omega_{M_g}([\gamma_1^i], [\gamma_2^i]) = 1$.*

Proof. For every $i \in [t-1]$, let $H^{i,0}$ be the union of the graph H^i with the unique xy-path P^0 in $F^0 + yw^0$. The graph $F^{i,0} = F^i \cup P^0$ is a spanning tree of $H^{i,0}$, and $E(H^{i,0}) \setminus E(F^{i,0}) = E(H^i) \setminus E(T)$.

Let $\mathcal{E}$ be a drawing of G from Lemma 12. If $H = K_{3,3}$, we apply Corollary 23 to $G = H^i$ and $F = F^i$. If $H = K_5 - e$ or $H = K_{3,3} - e$ where $e = xy$, we apply Corollary 23 to $G = H^i + e$, $F = F^i + e$, and the drawing of $H^i + e$ where e is drawn along the path P^0 in $\mathcal{E}$ (with self-crossings removed if necessary). In each of the three cases at least one of the following alternatives occurs:

(1) $y_{e_j^i}^\top y_{g_{j'}^i} = 1$ for some $j \in [k]$ and $j' \in [m]$,
(2) $y_{f_j^i}^\top y_{g_{j'}^i} = 1$ for some $j \in [l]$ and $j' \in [m]$,
(3) $y_{h^i}^\top y_{g_{j'}^i} = 1$ for some $j' \in [m]$,
(4) $y_{g_{j'}^i}^\top y_{g_{j''}^i} = 1$ for some $j', j'' \in [m]$,
(5) $y_{e_j^i}^\top (y_{f_{j'}^i} + y_{f_{j''}^i}) = 1$ for some $j \in [k]$ and $j', j'' \in [l]$,
(6) $y_{e_j^i}^\top (y_{f_{j'}^i} + y_{h^i}) = 1$ for some $j \in [k]$ and $j' \in [l]$.

Here we used Observation 25 for each $j \in [k]$ to pair the edges of $E(H^i) \setminus E(T)$ incident with y and independent from e_j^i. We note that in each of the six alternatives the edges on the left side of the scalar product can be required to be independent from the edges on the right side; however, we do not use this fact in further arguments.

To finish the proof of the lemma for $i \in [t-1]$, we use Observation 19 together with the additional fact that for every $j', j'' \in [l]$, the cycle $C_{f_{j'}^i} + C_{f_{j''}^i}$ is a subgraph of $H^i - x$. In

particular, in case 1) we choose $C_1^i = C_{g^i_{j'}}$ and $C_2^i = C_{e^i_j}$, in case 2) we choose $C_1^i = C_{f^i_j}$ and $C_2^i = C_{g^i_{j'}}$, in case 3) we choose $C_1^i = C_{h^i}$ and $C_2^i = C_{g^i_{j'}}$, in case 4) we choose $C_1^i = C_{g^i_{j'}}$ and $C_2^i = C_{g^i_{j''}}$, in case 5) we choose $C_1^i = C_{f^i_{j'}} + C_{f^i_{j''}}$ and $C_2^i = C_{e^i_j}$, and in case 6) we choose $C_1^i = C_{f^i_{j'}} + C_{h^i}$ and $C_2^i = C_{e^i_j}$.

Finally, by exchanging the roles of H^1 and H^0 in $\Pi_{x,y}tH$ in the proof, we also obtain cycles C_1^0 and C_2^0 with the required properties. ◀

We are now ready to finish the proof of Theorem 7b).

▶ **Proposition 27.** *Let $t \geq 2$ and let H be a Kuratowski xy-wing. Then $\mathrm{g}_0(\Pi_{x,y}tH) \geq \lceil t/2 \rceil$.*

Proof. Let $\mathcal{D}$ be an independently even drawing of $\Pi_{x,y}tH$ on M_g. For every $i \in [0, t-1]$, let C_1^i and C_2^i be the cycles from Lemma 26 and let γ_1^i and γ_2^i, respectively, be the closed curves representing them in $\mathcal{D}$.

Without loss of generality, we assume that there is an $s \in [0, t-1]$ such that

- for every $i \in [0, s]$, C_1^i is a subgraph of $H^i - x$ and C_2^i is a subgraph of $H^i - y$, and
- for every $i \in [s+1, t-1]$, the cycle C_2^i is a subgraph of $H^i - x - y$.

It follows that for distinct $i, i' \in [0, t]$, the cycles C_1^i and $C_2^{i'}$ are vertex-disjoint whenever $i, i' \in [0, s]$, $i, i' \in [s+1, t-1]$, or $i \leq s < i'$.

Let A be the $t \times t$ matrix with entries

$$A_{i,i'} = \Omega_{M_g}([\gamma_1^i], [\gamma_2^{i'}]).$$

By Lemma 26, Observation 11 and the previous discussion, the matrix A has 1-entries on the diagonal and 0-entries above the diagonal. Thus, the rank of A over $\mathbb{Z}_2$ is t. Hence, the rank of Ω_{M_g} is at least t, which implies $2g \geq t$. ◀

References

1. Joseph Battle, Frank Harary, Yukihiro Kodama, and J. W. T. Youngs. Additivity of the genus of a graph. *Bull. Amer. Math. Soc.*, 68:565–568, 1962. doi:10.1090/S0002-9904-1962-10847-7.
2. Thomas Böhme, Ken-ichi Kawarabayashi, John Maharry, and Bojan Mohar. $K_{3,k}$-minors in large 7-connected graphs. Preprint available at http://preprinti.imfm.si/PDF/01051.pdf, 2008.
3. Thomas Böhme, Ken-ichi Kawarabayashi, John Maharry, and Bojan Mohar. Linear connectivity forces large complete bipartite minors. *J. Combin. Theory Ser. B*, 99(3):557–582, 2009. doi:10.1016/j.jctb.2008.07.006.
4. André Bouchet. Orientable and nonorientable genus of the complete bipartite graph. *J. Combin. Theory Ser. B*, 24(1):24–33, 1978. doi:10.1016/0095-8956(78)90073-4.
5. G. Cairns and Y. Nikolayevsky. Bounds for generalized thrackles. *Discrete Comput. Geom.*, 23(2):191–206, 2000. doi:10.1007/PL00009495.
6. R. Christian, R. B. Richter, and G. Salazar. Embedding a graph-like continuum in some surface. *J. Graph Theory*, 79(2):159–165, 2015. doi:10.1002/jgt.21823.
7. Éric Colin de Verdière, Vojtěch Kaluža, Pavel Paták, Zuzana Patáková, and Martin Tancer. A direct proof of the strong Hanani–Tutte theorem on the projective plane. *J. Graph Algorithms Appl.*, 21(5):939–981, 2017. doi:10.7155/jgaa.00445.
8. D. de Caen. The ranks of tournament matrices. *Amer. Math. Monthly*, 98(9):829–831, 1991. doi:10.2307/2324270.
9. R. W. Decker, H. H. Glover, and J. P. Huneke. Computing the genus of the 2-amalgamations of graphs. *Combinatorica*, 5(4):271–282, 1985. doi:10.1007/BF02579241.

10 J. R. Fiedler, J. P. Huneke, R. B. Richter, and N. Robertson. Computing the orientable genus of projective graphs. *J. Graph Theory*, 20(3):297–308, 1995. doi:10.1002/jgt.3190200305.

11 R. Fulek and J. Kynčl. Counterexample to an extension of the hanani–tutte theorem on the surface of genus 4. Submitted, arXiv:1709.00508, 2017.

12 R. Fulek and J. Kynčl. The $\mathbb{Z}_2$-genus of kuratowski minors. Manuscript, ???, 2018.

13 Haim Hanani. Über wesentlich unplättbare Kurven im drei-dimensionalen Raume. *Fundamenta Mathematicae*, 23:135–142, 1934.

14 Allen Hatcher. *Algebraic topology*. Cambridge University Press, Cambridge, 2002.

15 D. J. Kleitman. A note on the parity of the number of crossings of a graph. *J. Combinatorial Theory Ser. B*, 21(1):88–89, 1976.

16 Bojan Mohar and Carsten Thomassen. *Graphs on surfaces*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, 2001.

17 Michael J. Pelsmajer, Marcus Schaefer, and Despina Stasi. Strong Hanani–Tutte on the projective plane. *SIAM J. Discrete Math.*, 23(3):1317–1323, 2009. doi:10.1137/08072485X.

18 Michael J. Pelsmajer, Marcus Schaefer, and Daniel Štefankovič. Removing even crossings on surfaces. *European J. Combin.*, 30(7):1704–1717, 2009. doi:10.1016/j.ejc.2009.03.002.

19 Gerhard Ringel. Das Geschlecht des vollständigen paaren Graphen. *Abh. Math. Sem. Univ. Hamburg*, 28:139–150, 1965. doi:10.1007/BF02993245.

20 Neil Robertson and Richard Vitray. Representativity of surface embeddings. In *Paths, flows, and VLSI-layout (Bonn, 1988)*, volume 9 of *Algorithms Combin.*, pages 293–328. Springer, Berlin, 1990.

21 Marcus Schaefer. Hanani-Tutte and related results. In *Geometry—intuitive, discrete, and convex*, volume 24 of *Bolyai Soc. Math. Stud.*, pages 259–299. János Bolyai Math. Soc., Budapest, 2013. doi:10.1007/978-3-642-41498-5_10.

22 Marcus Schaefer and Daniel Štefankovič. Block additivity of $\mathbb{Z}_2$-embeddings. In *Graph drawing*, volume 8242 of *Lecture Notes in Comput. Sci.*, pages 185–195. Springer, Cham, 2013. doi:10.1007/978-3-319-03841-4_17.

23 Paul Seymour, 2017. Personal communication.

24 W. T. Tutte. Toward a theory of crossing numbers. *J. Combinatorial Theory*, 8:45–53, 1970.

Shellability is NP-Complete

Xavier Goaoc[1]
LIGM, Université Paris-Est
UMR 8049 CNRS, ENPC, ESIEE, UPEM, F-77454, Marne-la-Vallée, France
xavier.goaoc@u-pem.fr

Pavel Paták
Department of Mathematics and Statistics, Masaryk University
Brno, Czech Republic
patak@math.muni.cz
https://orcid.org/0000-0003-3016-0278

Zuzana Patáková
IST Austria
Klosterneuburg, Austria
zuzka@kam.mff.cuni.cz
https://orcid.org/0000-0002-3975-1683

Martin Tancer[2]
Department of Applied Mathematics, Charles University
Prague, Czech Republic
tancer@kam.mff.cuni.cz

Uli Wagner
IST Austria
Klosterneuburg, Austria
uli@ist.ac.at
https://orcid.org/0000-0002-1494-0568

Abstract

We prove that for every $d \geq 2$, deciding if a pure, d-dimensional, simplicial complex is shellable is NP-hard, hence NP-complete. This resolves a question raised, e.g., by Danaraj and Klee in 1978. Our reduction also yields that for every $d \geq 2$ and $k \geq 0$, deciding if a pure, d-dimensional, simplicial complex is k-decomposable is NP-hard. For $d \geq 3$, both problems remain NP-hard when restricted to contractible pure d-dimensional complexes.

2012 ACM Subject Classification Mathematics of computing → Geometric topology, Theory of computation → Computational geometry

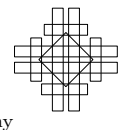
Keywords and phrases Shellability, simplicial complexes, NP-completeness, collapsibility

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.41

Related Version A full version of the paper is available at [14], `https://arxiv.org/abs/1711.08436`.

Funding Partially supported by the project EMBEDS II (CZ: 7AMB17FR029, FR: 38087RM) of Czech-French collaboration.

[1] Partially supported by IUF.
[2] Partially supported by the GAČR grant 16-01602Y and by Charles University project UNCE/SCI/004.



Acknowledgements We would like to thank Andrés D. Santamaría Galvis, Russ Woodroofe and anonymous referees for useful remarks on a preliminary version on this paper.

1 Introduction

A d-dimensional simplicial complex is called *pure* if all its facets (i.e., inclusion-maximal faces) have the same dimension d. A pure d-dimensional simplicial complex is *shellable* if there exists a linear ordering $\sigma_1, \sigma_2, \ldots, \sigma_n$ of its facets such that, for every $i \geq 2$, $\sigma_i \cap (\cup_{j<i}\sigma_j)$ is a pure $(d-1)$-dimensional simplicial complex; such an ordering is called a *shelling* or *shelling order*. For example, the boundary of a simplex is shellable (any order works), but no triangulation of the torus is (the condition fails for the first triangle σ_i that creates a non-contractible 1-cycle).

The concept of shellings originated in the theory of convex polytopes (in a more general version for polytopal complexes), as an inductive procedure to construct the boundary of a polytope by adding the facets one by one in such a way that all intermediate complexes (except the last one) are contractible. The fact that this is always possible, i.e., that convex polytopes are shellable, was initially used as an unproven assumption in early papers (see the discussion in [15, pp. 141–142] for a more detailed account of the history), before being proved by Bruggesser and Mani [9].

The notion of shellability extends to more general objects (including non-pure simplicial complexes and posets [8]), and plays an important role in diverse areas including piecewise-linear topology [26, 3], polytope theory (e.g., McMullen's proof of the *Upper Bound Theorem* [22]), topological combinatorics [5], algebraic combinatorics and commutative algebra [28, 24] , poset theory , and group theory [4, 27]; for a more detailed introduction and further references see [31, §3].

One of the reasons for its importance is that shellability – a combinatorial property – has strong topological implications: For example, if a pure d-dimensional complex K is a *pseudomanifold*[3] – which can be checked in linear time – and shellable, then K is homeomorphic to the sphere S^d (or the ball B^d, in case K has nonempty boundary) [10] – a property that is algorithmically undecidable for $d \geq 5$, by a celebrated result of Novikov [30, 23]. More generally, every pure d-dimensional shellable complex is homotopy equivalent to a wedge of d-spheres, in particular it is $(d-1)$-connected.

1.1 Results

From a computational viewpoint, it is natural to ask if one can decide efficiently (in polynomial time) whether a given complex is shellable. This question was raised at least as early as in the 1970's [11, 12] (see also [18, Problem 34]) and is of both practical and theoretical importance (besides direct consequences for the experimental exploration of simplicial complexes, the answer is also closely related to the question there are simple conditions that would characterize shellability). Danaraj and Klee proved that shellability of 2-dimensional pseudomanifolds can be tested in linear time [11], whereas a number of related problems have been shown to be NP-complete [13, 19, 17, 20, 29, 2] (see Section 1.2), but the computational complexity of the shellability problem has remained open. Here we show:[4]

[3] A pure, d-dimensional complex K is a pseudomanifold (with boundary) if every $(d-1)$-face of K is contained in exactly two (at most two) facets. (Sometimes, it is additionally required that the facet-adjacency graph of K is connected, but this does not matter in our setting, since shellable complexes always satisfy this connectivity property.)

[4] For basic notions from computational complexity, such as NP-completeness or reductions, see, e.g., [1].

▶ **Theorem 1.** *Deciding if a pure* 2*-dimensional simplicial complex is shellable is* NP*-complete.*

Here, the input is given as a finite abstract simplicial complex (see Section 2). [5]

▶ Remark. The problem of testing shellability is easily seen to lie in the complexity class NP (given a linear ordering of the facets of a complex, it is straightforward to check whether it is a shelling). Thus, the nontrivial part of Theorem 1 is that deciding shellability of pure 2-dimensional complexes is NP-hard.

It is easy to check that a pure simplicial complex K is shellable if and only if the *cone* $\{v\} * K$ is shellable, where v is a vertex not in K (see Section 2). Thus, the hardness of deciding shellability easily propagates to higher-dimensional complexes, even to cones.

▶ **Corollary 2.** *For $d \geq 3$, deciding if a pure d-dimensional complex is shellable is* NP*-complete even when the input is assumed to be a cone (hence contractible).*

Moreover, our hardness reduction (from 3-SAT) used in the proof of Theorem 1 (see Section 3) turns out to be sufficiently robust to also imply hardness results for a number of related problems.

Hardness of k-decomposability and CL-shellability. Let $d \geq 2$ and $k \geq 0$. A pure d-dimensional simplicial complex K is *k-decomposable* if it is a simplex or if there exists a face σ of K of dimension at most k such that (i) the link of σ in K is pure $(d - |\sigma|)$-dimensional and k-decomposable, and (ii) deleting σ and faces of K containing σ produces a d-dimensional k-decomposable complex. This notion, introduced by Provan and Billera [25], provides a hierarchy of properties (k-decomposability implies $(k+1)$-decomposability) interpolating between *vertex-decomposable* complexes ($k = 0$) and shellable complexes (shellability is equivalent to d-decomposability [25]). The initial motivation for considering this hierarchy was to study the *Hirsch conjecture* on combinatiorial diameters of convex polyhedra, or in the language of simplicial complex, the diameter of the facet-adjacency graphs of pure simplicial complexes: at one end, the boundary complex of every d-dimensional simplicial polytope is shellable [9], and at the other end, every vertex-decomposable simplicial complex has small diameter (it satisfies the *Hirsch bound* [25]).

▶ **Theorem 3.** *Let $d \geq 2$ and $k \geq 0$. Deciding if a pure d-dimensional simplicial complex is k-decomposable is* NP*-hard. For $d \geq 3$, the problem is already* NP*-hard for pure d-dimensional simplicial complexes that are cones (hence contractible).*

Another notion related to shellability is the *CL-shellability* of a poset, introduced in [6]. We do not reproduce the definition here, but note that a simplicial complex is shellable if and only if the dual of its face lattice is CL-shellable [7, Corollary 4.4]. For any fixed dimension d the face lattice can be computed in polynomial time, so we get:

▶ **Corollary 4.** *Deciding whether a given poset is CL-shellable is* NP*-hard.*

[5] There are several different ways of encoding an abstract simplicial complex – e.g., we can list the facets, or we can list all of its simplices –, but since we work with complexes of fixed dimension, these encodings can be translated into one another in polynomial time, so the precise choice does not matter.

1.2 Related work on collapsibility and our approach

Our proof of Theorem 1 builds on earlier results concerning *collapsibility*, a combinatorial analogue, introduced by Whitehead [32], of the topological notion of contractibility.[6] A face σ of a simplicial complex K is *free* if there is a unique inclusion-maximal face τ of K with $\sigma \subsetneq \tau$. An *elementary collapse* is the operation of deleting a free face and all faces containing it. A simplicial complex K *collapses* to a subcomplex $L \subseteq K$ if L can be obtained from K by a finite sequence of elementary collapses; K is called *collapsible* if it collapses to a single vertex.

The problem of deciding whether a given 3-dimensional complex is collapsible is NP-complete [29]; the proof builds on earlier work of Malgouyres and Francés [20], who showed that it is NP-complete to decide whether a given 3-dimensional complex collapses to some 1-dimensional subcomplex. By contrast, collapsibility of 2-dimensional complexes can be decided in polynomial time (by a greedy algorithm) [17, 20]. It follows that for any *fixed* integer k, it can be decided in polynomial time whether a given 2-dimensional simplicial complex can be made collapsible by deleting at most k faces of dimension 2; by contrast, the latter problem is NP-complete if k is part of the input [13].[7]

Our reduction uses the gadgets introduced by Malgouyres and Francés [20] and reworked in [29] to prove NP-hardness of deciding collapsibility for 3-dimensional complexes. However, these gadgets are not pure: they contain maximal simplices of two different dimensions, 2 and 3. Roughly speaking, we fix this by replacing the 3-dimensional subcomplexes by suitably triangulated 2-spheres and modifying the way in which they are glued. Interestingly, this also makes our reduction robust to subdivision and applicable to other types of decomposition.

Collapsibility and shellability. Furthermore, we will use the following connection between shellability and collapsibility, due to Hachimori [16] (throughout, $\tilde{\chi}$ denotes the reduced Euler characteristic).

▶ **Theorem 5** ([16, Theorem 8]). *Let K be a 2-dimensional simplicial complex. The second barycentric subdivision $\mathrm{sd}^2 K$ is shellable if and only if the link of each vertex of K is connected and there exists $\tilde{\chi}(K)$ triangles in K whose removal makes K collapsible.*

At first glance, Hachimori's theorem might suggest to prove Theorem 1 by a direct polynomial-time reduction of collapsibility to shellability. However, for 2-dimensional complexes this would not imply hardness, since, as mentioned above, collapsibility of 2-dimensional complexes is decidable in polynomial time [17, 20]. Instead, we will use the existential part of Hachimori's theorem ("there exists $\tilde{\chi}(K)$ triangles") to encode instances of the 3-SAT problem, a classical NP-complete problem.

2 Notation and terminology

We give here an overview of the basic terminology, including the notions used but not defined in the introduction. We assume that the reader is familiar with standard concepts regarding simplicial complexes, and mostly list the notions we use and set up the notation.

[6] Collapsibility implies contractibility, but the latter property is undecidable for complexes of dimension at least 4 (this follows from Novikov's result [30], see [29, Appendix A]), whereas the problem of deciding collapsibility lies in NP.

[7] We remark that building on [13], a related problem, namely computing *optimal discrete Morse matchings* in simplicial complexes (which we will not define here), was also shown to be NP-complete [19, 17].

We recall that the input in Theorem 1 is assumed to be described as an abstract simplicial complex,[8] i.e., a purely combinatorial object. For the purposes of the exposition, however, it will be more convenient to use a description via geometric simplicial complexes.[9] In fact, in our construction, we will sometimes first describe a polyhedron[10] and only then a geometric simplicial complex triangulating the polyhedron, with the understanding that this is simply a convenient way to specify the associated abstract simplicial complex.

A *subdivision* of a (geometric) complex K is a complex K' such that the polyhedra of K and of K' coincide and every simplex of K' is contained in some simplex of K. The *reduced Euler characteristic* of a complex K is defined as $\tilde{\chi}(K) = \sum_{i=-1}^{\dim K} (-1)^i f_i(K)$ where $f_i(K)$ is the number of i-dimensional faces of K and, by convention, $f_{-1}(K)$ is 0 if K is empty and 1 otherwise.

For the definitions of *links*, the *barycentric subdivision*, $\operatorname{sd} K$, or the *join* $K * L$ of two complexes K and L we refer to the standard sources such as [21, Chapter 1] (or to to the full version [14]). We denote by Δ_ℓ the simplex of dimension ℓ.

3 The main proposition and its consequences

The cornerstone of our argument is the following construction:

▶ **Proposition 6.** *There is an algorithm that, given a* 3*-CNF formula*[11] ϕ*, produces, in time polynomial in the size of* ϕ*, a* 2*-dimensional simplicial complex* K_ϕ *with the following properties:*

- **(i)** *the link of every vertex of* K_ϕ *is connected,*
- **(ii)** *if* ϕ *is satisfiable, then* K_ϕ *becomes collapsible after removing some* $\tilde{\chi}(K_\phi)$ *triangles,*
- **(iii)** *if an arbitrary subdivision of* K_ϕ *becomes collapsible after removing some* $\tilde{\chi}(K_\phi)$ *triangles, then* ϕ *is satisfiable.*

The rest of this section derives our main result and its variants from Proposition 6. We then describe the construction of K_ϕ in Section 4 and prove Proposition 6 in Sections 5 and 6 (modulo a few claims, treated in detail in the full version [14]).

Hardness of shellability. Proposition 6 and Hachimori's theorem imply our main result:

Proof of Theorem 1. Let ϕ be a 3-CNF formula and let K_ϕ denote the 2-dimensional complex built according to Proposition 6. Since the link of every vertex of K_ϕ is connected, Theorem 5 guarantees that $\operatorname{sd}^2 K_\phi$ is shellable if and only if there exist $\tilde{\chi}(K_\phi)$ triangles whose removal makes K_ϕ collapsible. Hence, by statements (ii) and (iii), the formula ϕ is satisfiable

[8] A (finite) *abstract simplicial complex* is a collection K of subsets of a finite set V that is closed under taking subsets, i.e., if $\sigma \in K$ and $\tau \subseteq \sigma$, then $\tau \in K$. The elements $v \in V$ are called the *vertices* of K (and often identified with the singleton sets $\{v\} \in K$), and the elements of K are called *faces* or *simplices* of K. The *dimension* of a face is its cardinality minus 1, and the *dimension* of K is the maximum dimension of any face. This is a purely combinatorial description of a simplicial complex and a natural input model for computational questions.

[9] A (finite) *geometric simplicial complex* is a finite collection K of geometric simplices (convex hulls of affinely independent points) in $\mathbb{R}^d$ (for some d) such that (i) if $\sigma \in K$ and τ is a face of σ, then τ also belongs to K, and (ii) if $\sigma_1, \sigma_2 \in K$, then $\sigma_1 \cap \sigma_2$ is a face of both σ_1 and σ_2. There is a straightforward translation between the two descriptions (see, e.g. [21, Chapter 1]), and this is the setting we will work in for the rest of the article.

[10] The *polyhedron* of a geometric simplicial complex K is defined as the union of simplices contained in K, $\bigcup_{\sigma \in K} \sigma$. We also say that K *triangulates* $X \subseteq \mathbb{R}^d$ if X is the polyhedron of K. Note that a given polyhedron usually has many different triangulations.

[11] That is, a boolean formula in conjunctive normal form such that each clause consists of three literals.

if and only if $\mathrm{sd}^2 K_\phi$ is shellable. Taking the barycentric subdivision of a two-dimensional complex multiplies its number of simplices by at most a constant factor. The complex $\mathrm{sd}^2 K_\phi$ can thus be constructed from ϕ in polynomial time, and 3-SAT reduces in polynomial time to deciding the shellability of 2-dimensional pure complexes. ◀

Hardness of k-decomposability. Note that statement (iii) in Proposition 6 deals with arbitrary subdivision whereas we needed it without subdivisions in the proof above. This extra elbow room comes at no cost in the proof of Proposition 6 and yields the NP-hardness of k-decomposability.

Proof of Theorem 3. Assume without loss of generality that $k \leq d$. Let ϕ be a 3-CNF formula and K_ϕ the complex produced by Proposition 6. We have the following implications:[12]

$$
\begin{array}{lll}
\phi \text{ is satisfiable} & \Rightarrow & K_\phi \text{ is collapsible after removal of some } \tilde{\chi}(K_\phi) \text{ triangles} \\
& \Rightarrow & \mathrm{sd}^2 K_\phi \text{ is shellable} \\
& \Rightarrow_{(b)} & \mathrm{sd}^3 K_\phi \text{ is vertex-decomposable} \\
& \Rightarrow_{(c)} & \Delta_{d-3} * \mathrm{sd}^3 K_\phi \text{ is vertex-decomposable} \\
& \Rightarrow_{(a)} & \Delta_{d-3} * \mathrm{sd}^3 K_\phi \text{ is } k\text{-decomposable} \\
& \Rightarrow_{(a)} & \Delta_{d-3} * \mathrm{sd}^3 K_\phi \text{ is shellable (i.e., } d\text{-decomposable)} \\
& \Rightarrow_{(d)} & \mathrm{sd}^3 K_\phi \text{ is shellable} \\
& \Rightarrow & \mathrm{sd}\, K_\phi \text{ is collapsible after removal of some } \tilde{\chi}(K_\phi) \text{ triangles} \\
& \Rightarrow & \phi \text{ is satisfiable}
\end{array}
$$

The first and last implications are by construction of K_ϕ (Proposition 6). The second and second to last follow from Theorem 5, given that Proposition 6 ensures that links of vertices in K_ϕ are connected. The remaining implications follow from the following known facts:

(a) if K is k-decomposable, then K is k'-decomposable for $k' \geq k$,

(b) if K is shellable, then $\mathrm{sd}\, K$ is vertex-decomposable [8],

(c) K is vertex-decomposable if and only if $\Delta_\ell * K$ is vertex-decomposable [25, Prop. 2.4],

(d) K is shellable if and only if $\Delta_\ell * K$ is shellable (see the full version [14] for details).

Since the first and last statement are identical, these are all equivalences. In particular, ϕ is satisfiable if and only if $\Delta_{d-3} * \mathrm{sd}^3 K_\phi$ is k-decomposable. Since this complex can be computed in time polynomial in the size of K_ϕ, *i.e.*, polynomial in the size of ϕ, the first statement follows. Since $\Delta_{d-3} * \mathrm{sd}^3 K_\phi$ is contractible for $d \geq 3$, the second statement follows. ◀

4 Construction

We now define the complex K_ϕ mentioned in Proposition 6. This complex consists of several building blocks, called *gadgets*. We first give a "functional" outline of the gadgets (in Section 4.1), insisting on the properties that guided their design, before moving on to the details of their construction and gluing (Sections 4.2 and 4.3).

We use the notational convention that complexes that depend on a variable u are denoted with round brackets, *e.g.* $f(u)$, whereas complexes that depend on a literal are denoted with square brackets, *e.g.* $f[u]$ or $f[\neg u]$.

[12] In the case $d = 2$, we use the convention that $\Delta_{-1} * L = L$ for any simplicial complex L.

4.1 Outline of the construction

The gadgets forming K_ϕ are designed with two ideas in mind. First, every gadget, when considered separately, can only be collapsed starting in a few *special edges*. Next, the special edges of each gadgets are intended to be glued to other gadgets, so as to create dependencies in the flow of collapses: if an edge f of a gadget $\mathbf{G}$ is attached to a triangle of another gadget $\mathbf{G}'$, then $\mathbf{G}$ cannot be collapsed starting by f before some part of $\mathbf{G}'$ has been collapsed.

Variable gadgets. For every variable u we create a gadget $\mathbf{V}(u)$. This gadget has three special edges; two are associated, respectively, with TRUE and FALSE; we call the third one "unlocking". Overall, the construction ensures that any removal of $\tilde{\chi}(K_\phi)$ triangles from K_ϕ either frees exactly one of the edges associated with TRUE or FALSE in every variable gadget, or makes K_ϕ obviously non-collapsible. This relates the removal of triangles in K_ϕ to the assignment of variables in ϕ. We also ensure that part of each variable gadget remains uncollapsible until the special unlocking edge is freed.

Clause gadgets. For every clause $c = \ell_1 \vee \ell_2 \vee \ell_3$ we create a gadget $\mathbf{C}(c)$. This gadget has three special edges, one per literal ℓ_i. Assume that $\ell_i \in \{u, \neg u\}$. Then the special edge associated with ℓ_i is attached to $\mathbf{V}(u)$ so that it can be freed if and only if the triangle removal phase freed the special edge of $\mathbf{V}(u)$ associated with TRUE (if $\ell_i = u$) or with FALSE (if $\ell_i = \neg u$). This ensures that the gadget $\mathbf{C}(c)$ can be collapsed if and only if one of its literals was "selected" at the triangle removal phase.

Conjunction gadget. We add a gadget $\mathbf{A}$ with a single special edge, that is attached to every clause gadget. This gadget can be collapsed only after the collapse of every clause gadget has started (hence, if every clause contains a literal selected at the triangle removal phase). In turn, the collapse of $\mathbf{A}$ will free the unlocking special edge of every variable gadget, allowing to complete the collapse.

Notations. For any variable u, we denote the special edges of $\mathbf{V}(u)$ associated with TRUE and FALSE by, respectively, $f[u]$ and $f[\neg u]$; we denote the unlocking edge by $f(u)$. For every clause $c = \ell_1 \vee \ell_2 \vee \ell_3$, we denote by $f[\ell_i, c]$ the special edge of $\mathbf{C}(c)$ associated with ℓ_i. We denote by f_{and} the special edge of the conjunction gadget $\mathbf{A}$. The attachment of these edges are summarized in the table below.

gadget	special edges	attached to	freed by
$\mathbf{V}(u)$	$f[u]$	-	triangle deletion
	$f[\neg u]$	-	triangle deletion
	$f(u)$	$\mathbf{A}$	freeing f_{and}
$\mathbf{C}(u_2 \vee \neg u_4 \vee u_9)$	$f[u_2, c]$	$\mathbf{V}(u_2)$	freeing $f[u_2]$
	$f[\neg u_4, c]$	$\mathbf{V}(u_4)$	freeing $f[\neg u_4]$
	$f[u_9, c]$	$\mathbf{V}(u_9)$	freeing $f[u_9]$
$\mathbf{A}$	f_{and}	every clause gadget	collapsing all clause gadgets

Flow of collapses. Let us summarize the mechanism sketched above. Assume that ϕ is satisfiable, and consider a satisfying assignment. Remove the triangles from each $\mathbf{V}(u)$ so that the edge that becomes free is $f[u]$ if u was assigned TRUE, and $f[\neg u]$ otherwise. This will allow to collapse each clause gadget in order to make f_{and} free. Consequently, we will be

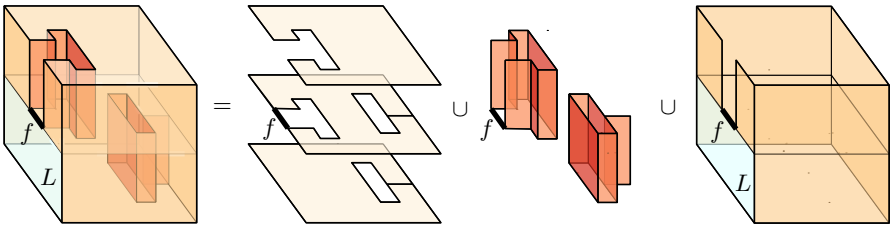
able to collapse $\mathbf{A}$ and make all unlocking edges $f(u)$ free. This allows finishing the collapses on all $\mathbf{V}(u)$.

On the other hand, to collapse K_ϕ we must collapse f_{and} at some point. Before this can happen, we have to collapse in each clause $c = \ell_1 \vee \ell_2 \vee \ell_3$ one of the edges $f[\ell_i, c]$. This, in turn, requires that $f[\ell_i]$ has been made free. If we can ensure that $f[\neg\ell_i]$ cannot also be free, then we can read off from the collapse an assignment of the variables that must satisfy every clause, and therefore ϕ. (If $\ell_i = u$, then we set u to TRUE, if $\ell_i = \neg u$, then we set u to FALSE. If there are unassigned variables after considering all clauses, we assign them arbitrarily.)

4.2 Preparation: modified Bing's houses

Our gadgets rely on two modifications of Bing's house, a classical example of a 2-dimensional simplicial complex that is contractible but not collapsible. Bing's house consists of a box split into two parts (or *rooms*); each room is connected to the outside by a tunnel through the other room; each tunnel is attached to the room that it traverses by a rectangle (or *wall*). The modifications that we use here make the complex collapsible, but restricts its set of free faces to exactly one or exactly three edges.

One free edge. We use here a modification due to Malgouyres and Francés [20]. In one of the rooms (say the top one), the wall has been thickened and hollowed out, see the figure below. We call the resulting polyhedron a Bing's house with a single free edge, or a 1-*house* for short. Two special elements of a 1-house are its *free edge* (denoted f and in thick stroke in the figure below) and its *lower wall* rectangle (denoted L and colored in light blue in the figure below). We only consider triangulations of 1-houses that subdivide the edge f and the lower wall L. We use 1-houses for the following properties:



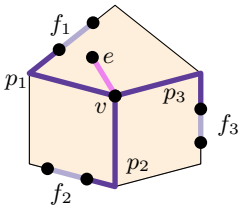
▶ **Lemma 7.** *Let B be a 1-house, f its free edge and L its lower wall. In any triangulation of B, the free faces are exactly the edges that subdivide f. Moreover, B collapses to any subtree of the 1-skeleton of B that is contained in L and shares with the boundary of L a single endpoint of f.*

The first statement follows from the fact that the edges that subdivide f are the only ones that are not part of two triangles; see [20, Remark 1]. The second statement was proven in [29, Lemma 7] for certain trees, but the argument holds for arbitrary trees; see the full version [14] for details.

When working with 1-houses, we will usually only describe the lower wall to clarify which subtree we intend to collapse to.

Three free edges. We also use the Bing's houses with three collapsed walls introduced in [29]; we call them 3-*houses* for short. These are 2-dimensional complexes whose construction is more involved; we thus state its main properties, so that we can use it as a black box, and

refer the reader interested in its precise definition to [29, §4]. Refer to the figure on the right (which corresponds to Figure 9 in [29]). The 3-house has exactly three free edges f_1, f_2, f_3, and has three distinguished paths p_1, p_2, p_3 sharing a common vertex v and such that each p_i shares exactly one vertex with f_i and no vertex with f_j for $j \neq i$. In addition, it contains an edge e incident to v so that the union of $p_1, p_2, p_3, f_1, f_2, f_3$ and e forms a subdivided star (in graph-theoretic sense) with four rays.



Let C denote the 3-house as described above. In [29], the polyhedron of C is described in detail but no triangulation is specified. We are happy with any concrete triangulation for which Lemma 8 below holds; we can in addition require that the paths p_1, p_2 and p_3 each consist of two edges.[13]

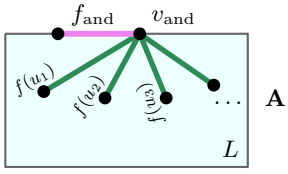
▶ **Lemma 8** ([29, Lemma 8]). *In any subdivision of C, the free faces are exactly the edges that subdivide f_1, f_2 and f_3. Moreover, C collapses to the 1-complex spanned by e, p_1, p_2, p_3 and any two of $\{f_1, f_2, f_3\}$.*

4.3 Detailed construction

Section 4.1 gave a quick description of the intended functions of the various gadgets. We now flesh them out and describe how they are glued together.

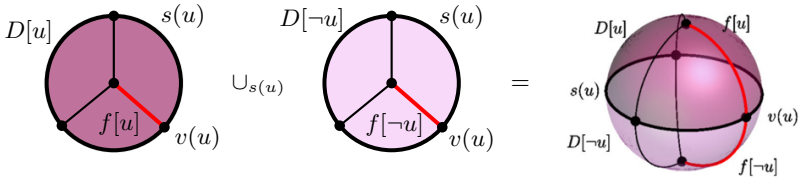
Triangulations. For some parts of the complex, it will be convenient to first describe the polyhedron, then discuss its triangulation. Our description of the triangulation may vary in precision: it may be omitted (if any reasonable triangulation works), given indirectly by the properties it should satisfy, or given explicitly (for instance to make it clear that we can glue the gadgets as announced).

Conjunction gadget. The conjunction gadget $\mathbf{A}$ is a 1-house. We let f_{and} denote its free edge and v_{and} one of the endpoints of f_{and}. We further triangulate the lower wall so that v_{and} has sufficiently high degree, allowing to assign every variable u to an internal edge $f(u)$ of the lower wall incident to v_{and}. See the lower left wall on the right picture. Any triangulation satisfying these prescriptions and computable in time polynomial in the size of ϕ suits our purpose.



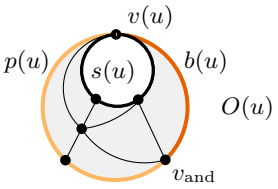
Variable gadget. The variable gadget $\mathbf{V}(u)$ associated with the variable u has four parts.

1. The first part is a triangulated 2-sphere $S(u)$ that consists of two disks $D[u]$ and $D[\neg u]$ sharing a common boundary circle $s(u)$. The circle $s(u)$ contains a distinguished vertex $v(u)$. The disk $D[u]$ (resp. $D[\neg u]$) has a distinguished edge $f[u]$ (resp. $f[\neg u]$) that joins $v(u)$ to its center.

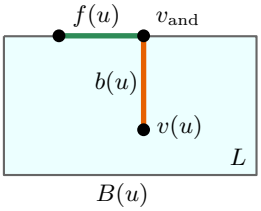


[13] The value two is not important here; what matters is to fix some value that can be used throughout the construction.

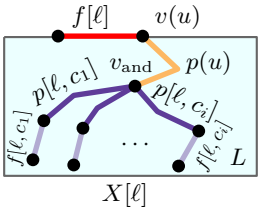
2. The second part is a 2-complex $O(u)$ that consists of two "boundary" circles sharing a vertex. The vertex is identified with the vertex $v(u)$ of $S(u)$. One of the circles is identified with $s(u)$. The other circle is decomposed into two arcs: one is a single edge named $b(u)$, the other is a path with two edges which we call $p(u)$. The vertex common to $b(u)$ and $p(u)$, distinct from $v(u)$, is identified with the vertex v_{and} of the conjunction gadget.



3. The third part is a 1-house $B(u)$ intended to block the edge $b(u) \in O(u)$ from being free as long as the conjunction gadget has not been collapsed. The free edge of $B(u)$ is identified with the edge $f(u)$ in the conjunction gadget $\mathbf{A}$ and the edge $b(u) \in O(u)$ is identified with an edge of the lower wall of $B(u)$ that shares the vertex v_{and} with $f(u)$.

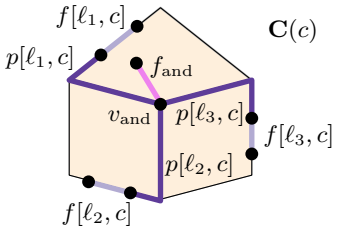


4. The fourth part consists of two complexes, $X[u]$ and $X[\neg u]$. Let $\ell \in \{u, \neg u\}$ and refer to the figure on the right. The complex $X[\ell]$ is a 1-house whose free edge is identified with the edge $f[\ell]$ from $D[\ell]$, and whose lower wall contains a path identified with $p(u)$. Hence, $p(u)$ is common to $X[u]$, $X[\neg u]$, and $O(u)$. For every clause c_i containing the literal ℓ, we add in the lower wall a two-edge path $p[\ell, c_i]$ extended by an edge $f[\ell, c_i]$; the path $p[\ell, c_i]$ intersects $p(u)$ in exactly v_{and} (in particular, these paths and edges form a subdivided star, in graph theoretic sense, centered at v_{and}).



Clause gadget. The clause gadget $\mathbf{C}(c)$ associated with the clause $c = \ell_1 \vee \ell_2 \vee \ell_3$ is a 3-house C where:

- the edges f_i of C are identified with the edges $f[\ell_i, c]$ in $X[\ell_i]$;
- the paths p_i of C are identified with the paths $p[\ell_i, c]$ in $X[\ell_i]$;
- the vertex v of C is identified with the vertex v_{and}; and
- the edge e of C is identified with the edge f_{and}.



Putting it all together. Let ϕ be a 3-CNF formula with variables $u_1, u_2, \ldots, u_n$ and clauses $c_1, c_2, \ldots, c_m$. The complex K_ϕ is defined as

$$K_\phi = \mathbf{A} \cup \left(\bigcup_{i=1}^{n} \underbrace{S(u_i) \cup O(u_i) \cup B(u_i) \cup X[u_i] \cup X[\neg u_i]}_{\mathbf{V}(u_i)} \right) \cup \left(\bigcup_{j=1}^{m} \mathbf{C}(c_j) \right).$$

4.4 Properties of K_ϕ

It remains to prove that K_ϕ has the properties required by Proposition 6. Item (i) is quite straightforward (although tedious) and it is proved in the full version [14]. Here we provide a sketch only.

Sketch of Proposition 6(i). We first check that the link of every vertex is connected within a 1-house, within a 3-house and within $S(u) \cup O(u)$ for every variable u. Then K_ϕ is obtained by gluing such subcomplexes together along edges which preserves the connectedness of the links. ◀

The proof of items (ii) and (iii) is given in the forthcoming sections. However, first we need to determine the reduced Euler characteristic of K_ϕ (again, we provide a sketch only; see the full version [14] for full proof):

▶ **Proposition 9.** *$\tilde{\chi}(K_\phi)$ equals the number of variables of ϕ.*

Sketch. Many of the gadgets are contractible. For example, if we replace a 1-house with its unique free edge, we do not affect the Euler characteristic. After a series of similar reductions we find out that the only contribution to the (reduced) Euler characteristic arises from the sphere $S(u)$, one for each variable. ◀

5 Satisfiability implies collapsibility

In this section we prove Proposition 6(ii), *i.e.* that if ϕ is satisfiable, then there exists a choice of $\tilde{\chi}(K_\phi)$ triangles of K_ϕ whose removal makes the complex collapsible.

Initial steps. Let us fix a satisfying assignment for ϕ. For every variable u, we set $\ell(u)$ to u if u is TRUE in our assignment, and to $\neg u$ otherwise. Next, for every variable u, we remove a triangle from the region $D[\ell(u)]$ of the sphere $S(u)$. Proposition 9 ensures that this removes precisely $\tilde{\chi}(K_\phi)$ triangles, as announced.

Constraint complex. It will be convenient to analyze collapses of K_ϕ locally within a subcomplex, typically a gadget. To do so formally, we use constraint complexes following [29]. Given a simplicial complex K and a subcomplex M of K, we define the *constraint complex* of (K, M), denoted $\Gamma(K, M)$, as $\Gamma(K, M) := \{\vartheta \in M \colon \exists \eta \in K \setminus M \text{ s.t. } \vartheta \subset \eta\}$.

▶ **Lemma 10** ([29, Lemma 4])**.** *Let K be a simplicial complex and M a subcomplex of K. If M collapses to M' and $\Gamma(K, M) \subseteq M'$ then K collapses to $(K \setminus M) \cup M'$.*

Collapses. We now describe a sequence of collapses enabled by the removal of the triangles. Recall that we started from the complex

$$K_\phi = \mathbf{A} \cup \left(\bigcup_{i=1}^{n} O(u_i) \cup S(u_i) \cup B(u_i) \cup X[u_i] \cup X[\neg u_i]\right) \cup \left(\bigcup_{j=1}^{m} \mathbf{C}(c_j)\right)$$

where $u_1, u_2, \ldots, u_n$ and $c_1, c_2, \ldots, c_m$ are, respectively, the variables and the clauses of ϕ. We then removed a triangle from each $D[\ell(u)]$. The removal of a triangle of $D[\ell(u)]$ allows to collapse that subcomplex to $s(u) \cup f[\ell(u)]$. This frees $f[\ell(u)]$. The complex becomes:

$$K_a = \mathbf{A} \cup \left(\bigcup_{i=1}^{n} O(u_i) \cup D[\neg\ell(u_i)] \cup B(u_i) \cup X[u_i] \cup X[\neg u_i]\right) \cup \left(\bigcup_{j=1}^{m} \mathbf{C}(c_j)\right).$$

We can then start to collapse the subcomplexes $X[\ell(u)]$. We proceed one variable at a time. Assume that we are about to proceed with the collapse of $X[\ell(u)]$ and let K denote the current complex. Locally, $X[\ell(u)]$ is a 1-house with free edge $f[\ell(u)]$. Moreover, $\Gamma(K, X[\ell(u)])$ is the tree $T(u)$ formed by the path $p(u)$ and the union of the paths $p[\ell(u), c] \cup f[\ell(u), c]$ for every clause c using the literal $\ell(u)$. Lemma 7 ensures that $X[\ell(u)]$ can be locally collapsed to $T(u)$, and Lemma 10 ensures that K can be globally collapsed to $(K \setminus X[\ell(u)]) \cup T(u)$. We proceed in this way for every complex $X[\ell(u)]$. The complex becomes:

$$K_b = \mathbf{A} \cup \left(\bigcup_{i=1}^{n} O(u_i) \cup D[\neg\ell(u_i)] \cup B(u_i) \cup X[\neg\ell(u_i)]\right) \cup \left(\bigcup_{j=1}^{m} \mathbf{C}(c_j)\right).$$

The collapses so far have freed every edge of $f[\ell(u_i), c]$. We now consider every clause c_j in turn. Put $c_j = (\ell_1 \vee \ell_2 \vee \ell_3)$ and let K denote the current complex. The assignment that we

chose is satisfying, so at least one of ℓ_1, ℓ_2 or ℓ_3 coincides with $\ell(u_i)$ for some i; let us assume without loss of generality that $\ell_1 = \ell(u_i)$. The edge $f[\ell_1, c]$ is therefore free and Lemma 8 yields that locally, $\mathbf{C}(c_j)$ collapses to the tree $T(c_j) = f_{\text{and}} \cup p[\ell_1, c] \cup p[\ell_2, c] \cup p[\ell_2, c] \cup f[\ell_2, c] \cup f[\ell_3, c]$. Moreover, $\Gamma(K, \mathbf{C}(c_j)) = T(c_j)$ so Lemma 10 ensure that K can be globally collapsed to $(K \setminus \mathbf{C}(c_j)) \cup T(c_j)$. After proceeding in this way for every complex $\mathbf{C}(c_j)$, we get:

$$K_c = \mathbf{A} \cup \left(\bigcup_{i=1}^{n} O(u_i) \cup D[\neg\ell(u_i)] \cup B(u_i) \cup X[\neg\ell(u_i)]\right) \cup \left(\bigcup_{j=1}^{m} T(c_j)\right).$$

The collapses so far have freed the edge f_{and}. We can then proceed to collapse $\mathbf{A}$. Locally, Lemma 7 allows to collapse $\mathbf{A}$ to the tree $T = f(u_1) \cup f(u_2) \cup \ldots \cup f(u_n)$. (From this point, we expect the reader to be able to check by her/himself that Lemma 10 allows to perform globally the collapse described locally.) The complex becomes:

$$K_d = \left(\bigcup_{i=1}^{n} O(u_i) \cup D[\neg\ell(u_i)] \cup B(u_i) \cup X[\neg\ell(u_i)]\right) \cup \left(\bigcup_{j=1}^{m} T(c_j)\right).$$

The collapses so far have freed every edge $f(u_i)$. Thus, Lemma 7 allows to collapse each complex $B(u_i)$ to its edge $b(u_i)$. This frees the edge $b(u_i)$, so the complex $O(u_i)$ can in turn be collapsed to $s(u_i) \cup p(u_i)$. At this point, the complex is:

$$K_e = \left(\bigcup_{i=1}^{n} s(u_i) \cup p(u_i) \cup D[\neg\ell(u_i)] \cup X[\neg\ell(u_i)]\right) \cup \left(\bigcup_{j=1}^{m} T(c_j)\right).$$

The collapses so far have freed every edge $s(u_i)$. We can thus collapse each $D[\neg\ell(u_i)]$ to $f[\neg\ell(u_i)]$. This frees every edge $f[\neg\ell(u_i)]$, allowing to collapse every subcomplex $X[\neg\ell(u_i)]$, again by Lemma 7, to the tree formed by the path $p(u_i)$ and the union of the paths $p[\neg\ell(u_i), c] \cup f[\neg\ell(u_i), c]$ for every clause c using the literal $\neg\ell(u_i)$.

At this point, we are left with a 1-dimensional complex. This complex is a tree (more precisely a subdivided star, in graph-theoretic sense, centered in v_{and} and consisting of the paths $p(u_i)$, the paths $p[\ell, c]$ and some of the edges $f[\ell, c]$). As any tree is collapsible, this completes the proof of Proposition 6(ii).

6 Collapsibility implies satisfiability

In this section we prove Proposition 6(iii), *i.e.* we consider some arbitrary subdivision K'_ϕ of K_ϕ, and prove that if K'_ϕ becomes collapsible after removing some $\tilde{\chi}(K_\phi)$ triangles, then ϕ is satisfiable. We thus consider a collapsible subcomplex $\widehat{K}$ of K'_ϕ obtained by removing $\tilde{\chi}(K_\phi)$ triangles from K'_ϕ.

Notations. Throughout this section, we use the following conventions. In general, we use hats (for example $\widehat{K}$) to denote subcomplexes of K'_ϕ. Given a subcomplex M of K_ϕ, we also write M' for the subcomplex of K'_ϕ that subdivides M.

Variable assignment from triangle removal. We first read our candidate assignment from the triangle removal following the same idea as in Section 5. This relies on two observations:

- The set of triangles removed in $\widehat{K}$ contains exactly one triangle from each sphere $S'(u)$. Indeed, since $\widehat{K}$ is collapsible and 2-dimensional, it cannot contain a 2-dimensional sphere. Hence, every sphere $S'(u)$ had at least one of its triangles removed. By Proposition 9, $\chi(K_\phi) = \chi(K'_\phi)$ equals the number of variables of ϕ, so this accounts for all removed triangles.
- For any variable u, any removed triangle in $S'(u)$ is either in $D'[u]$ or in $D'[\neg u]$. We give u the TRUE assignment in the former case and the FALSE assignment in the latter case.

The remainder of this section is devoted to prove that this assignment satisfies ϕ. It will again be convenient to denote by $\ell(u)$ the literal corresponding to this assignment, that is, $\ell(u) = u$ if u was assigned TRUE and $\ell(u) = \neg u$ otherwise.

Analyzing the collapse. Let us fix some collapse of $\widehat{K}$. We argue that our assignment satisfies ϕ by showing that these collapses must essentially follow the logical order of the collapse constructed in Section 5. To analyze the dependencies in the collapse, it is convenient to consider the partial order that it induces on the simplices of $\widehat{K}$: $\sigma \prec \tau$ if and only if in our collapse, σ is deleted before τ. We also write $\sigma \prec \widehat{M}$ for a subcomplex $\widehat{M}$ of $\widehat{K}$ if σ was removed before removing any simplex of $\widehat{M}$.

The key observation is the following dependency:

▶ **Lemma 11.** *There exists an edge $\widehat{e}$ of $\mathbf{A}'$ such that $\widehat{e} \prec D'[\neg\ell(u)]$ for every variable u.*

Proof. We first argue that for every variable u, there exists an edge $\widehat{e}_1(u) \in b'(u) \cup p'(u)$ such that $\widehat{e}_1(u) \prec D'[\neg\ell(u)]$. To see this, remark that the complex $D'[\neg\ell(u)] \cup O'(u)$ is fully contained in $\widehat{K}$ since the triangle removed from $S'(u)$ belongs to $D'[\ell(u)]$. It thus has to be collapsed. Since this complex is a disk, the first elementary collapse in $D'[\neg\ell(u)] \cup O'(u)$ has to involve some edge $\widehat{e}_1(u)$ of its boundary. This boundary is $b'(u) \cup p'(u)$, so it contains no edge of $D'[\neg\ell(u)]$. It follows that $\widehat{e}_1(u) \prec D'[\neg\ell(u)]$.

We next claim that $\widehat{e}_1(u) \in b'(u)$. Indeed, remark that every edge in $p'(u)$ belongs to two triangles of $X'[\neg\ell(u)]$. By Lemma 7, any collapse of $X'[\neg\ell(u)]$ must start by an elementary collapse using an edge of $f'[\neg\ell(u)]$ as a free face. Any edge of $f'[\neg\ell(u)]$ is, however, contained in two triangles of $D'[\neg\ell(u)]$ and thus cannot precede $D'[\neg\ell(u)]$ in $\prec$. It follows that $\widehat{e}_1(u) \in b'(u)$.

We can now identify $\widehat{e}$. Observe that $b'(u) \subset B'(u)$. As $B'(u)$ is a 1-house, Lemma 7 ensures that the first edge removed from $B'(u)$ must subdivide $f'(u)$. Hence, there is an edge $\widehat{e}_2(u) \subset f'(u)$ such that $\widehat{e}_2(u) \prec \widehat{e}_1(u)$. Since $f'(u) \subset \mathbf{A}'$, another 1-house, the same reasoning yields an edge $\widehat{e}_3(u)$ in f'_{and} such that $\widehat{e}_3(u) \prec \widehat{e}_2(u)$. Let $\widehat{e}$ denote the first edge removed from $\mathbf{A}'$ among all edges $\widehat{e}_3(u)$. At this point, we have for every variable u $\widehat{e} \prec \widehat{e}_2(u) \prec \widehat{e}_1(u) \prec D'[\neg\ell(u)]$, as announced. ◀

Let $\widehat{e}$ denote the edge of $\mathbf{A}'$ provided by Lemma 11, *i.e.* satisfying $\widehat{e} \prec D'[\neg\ell(u)]$ for every variable u. We can now check that the variable assignment does satisfy the formula:

- Consider a clause $c = (\ell_1 \vee \ell_2 \vee \ell_3)$ in ϕ. The complex $\mathbf{C}'(c)$ is a 3-house, so Lemma 8 restricts its set of free edges to the $f'[\ell_i, c]$. Hence, there is $i \in \{1, 2, 3\}$ and an edge $\widehat{e}_4(c)$ in $f'[\ell_i, c]$ such that $\widehat{e}_4(c) \preceq \mathbf{C}'(c)$. Note that, in particular, $\widehat{e}_4(c) \prec \widehat{e}$ as the edge f_{and} also belongs to $\mathbf{C}(c)$ and must be freed before collapsing $\mathbf{A}'$ (by Lemma 7).
- The subcomplex $f'[\ell_i, c]$ is contained not only in $\mathbf{C}'(c)$, but also in $X[\ell_i]$ which is a 1-house with free edge $f[\ell_i]$. By Lemma 7, the first elementary collapse of $X[\ell_i]$ uses as free face an edge $\widehat{e}_5(c)$ that subdivides $f'[\ell_i]$. In particular, $\widehat{e}_5(c) \prec f'[\ell_i, c]$ and $\widehat{e}_5(c) \prec \widehat{e}_4(c)$.
- Let u be the variable of the literal ℓ_i, that is, $\ell_i = u$, or $\ell_i = \neg u$; in particular $\ell_i \in \{\ell(u), \neg\ell(u)\}$. From $\widehat{e}_5(c) \prec \widehat{e}_4(c) \prec \widehat{e} \prec D'[\neg\ell(u)]$ it comes that $\widehat{e}_5(c)$ cannot belong to $D'[\neg\ell(u)]$. Yet, $\widehat{e}_5(c)$ belongs to $f'[\ell_i]$. It follows that $\ell_i \neq \neg\ell(u)$ and we must have $\ell_i = \ell(u)$. The definition of $\ell(u)$ thus ensures that our assignment satisfies the clause c.

Since our assignment satisfies every clause, it satisfies ϕ.

References

1 S. Arora and B. Barak. *Complexity Theory: A Modern Approach*. Cambridge University Press, Cambridge, 2009. URL: `http://www.cs.princeton.edu/theory/complexity/`.

2 D. Attali, O. Devillers, M. Glisse, and S. Lazard. Recognizing shrinkable complexes is NP-complete. *Journal of Computational Geometry*, 7(1):430–443, 2016.

3 R. H. Bing. *The geometric topology of 3-manifolds*, volume 40 of *American Mathematical Society Colloquium Publications*. American Mathematical Society, Providence, RI, 1983. doi:10.1090/coll/040.

4 A. Björner. Shellable and Cohen-Macaulay partially ordered sets. *Trans. Amer. Math. Soc.*, 260(1):159–183, 1980. doi:10.2307/1999881.

5 A. Björner. Topological methods. In *Handbook of combinatorics, Vol. 1, 2*, pages 1819–1872. Elsevier Sci. B. V., Amsterdam, 1995.

6 A. Björner and M. Wachs. Bruhat order of Coxeter groups and shellability. *Advances in Mathematics*, 43(1):87–100, 1982. doi:10.1016/0001-8708(82)90029-9.

7 A. Björner and M. Wachs. On lexicographically shellable posets. *Trans. Amer. Math. Soc.*, 277(1):323–341, 1983. doi:10.2307/1999359.

8 A. Björner and M. L. Wachs. Shellable nonpure complexes and posets. II. *Trans. Amer. Math. Soc.*, 349(10):3945–3975, 1997. doi:10.1090/S0002-9947-97-01838-2.

9 H. Bruggesser and P. Mani. Shellable decompositions of cells and spheres. *Mathematica Scandinavica*, 29(2):197–205, 1972.

10 G. Danaraj and V. Klee. Shellings of spheres and polytopes. *Duke Mathematical Journal*, 41(2):443–451, 1974.

11 G. Danaraj and V. Klee. A representation of 2-dimensional pseudomanifolds and its use in the design of a linear-time shelling algorithm. *Ann. Discrete Math.*, 2:53–63, 1978. Algorithmic aspects of combinatorics (Conf., Vancouver Island, B.C., 1976).

12 G. Danaraj and V. Klee. Which spheres are shellable? *Ann. Discrete Math.*, 2:33–52, 1978. Algorithmic aspects of combinatorics (Conf., Vancouver Island, B.C., 1976).

13 Ö. Eğecioğlu and T. F. Gonzalez. A computationally intractable problem on simplicial complexes. *Comput. Geom.*, 6(2):85–98, 1996. doi:10.1016/0925-7721(95)00015-1.

14 X. Goaoc, P. Paták, Z. Patáková, M. Tancer, and U. Wagner. Shellability is NP-complete, 2018. Preprint, https://arxiv.org/abs/1711.08436.

15 B. Grünbaum. *Convex polytopes*, volume 221 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, second edition, 2003. Prepared and with a preface by Volker Kaibel, Victor Klee and Günter M. Ziegler. doi:10.1007/978-1-4613-0019-9.

16 M. Hachimori. Decompositions of two-dimensional simplicial complexes. *Discrete Math.*, 308(11):2307–2312, 2008.

17 M. Joswig and M. E. Pfetsch. Computing optimal Morse matchings. *SIAM Journal on Discrete Mathematics*, 20(1):11–25, 2006.

18 V. Kaibel and M. E. Pfetsch. Some algorithmic problems in polytope theory. In *Algebra, geometry, and software systems*, pages 23–47. Springer, Berlin, 2003.

19 T. Lewiner, H. Lopes, and G. Tavares. Optimal discrete Morse functions for 2-manifolds. *Comput. Geom.*, 26(3):221–233, 2003. doi:10.1016/S0925-7721(03)00014-2.

20 R. Malgouyres and A. R. Francés. Determining whether a simplicial 3-complex collapses to a 1-complex is NP-complete. *DGCI*, pages 177–188, 2008.

21 J. Matoušek. *Using the Borsuk-Ulam theorem*. Universitext. Springer-Verlag, Berlin, 2007.

22 P. McMullen. The maximum numbers of faces of a convex polytope. *Mathematika*, 17(2):179–184, 1970.

23 A. Nabutovsky. Einstein structures: Existence versus uniqueness. *Geom. Funct. Anal.*, 5(1):76–91, 1995.

24 I. Peeva, V. Reiner, and B. Sturmfels. How to shell a monoid. *Math. Ann.*, 310(2):379–393, 1998.

25 J. S. Provan and L. J. Billera. Decompositions of simplicial complexes related to diameters of convex polyhedra. *Math. Oper. Res.*, 5(4):576–594, 1980. doi:10.1287/moor.5.4.576.

26 C. P. Rourke and B. J. Sanderson. *Introduction to piecewise-linear topology*. Springer Study Edition. Springer-Verlag, Berlin-New York, 1982. Reprint.

27 J. Shareshian. On the shellability of the order complex of the subgroup lattice of a finite group. *Trans. Amer. Math. Soc.*, 353(7):2689–2703, 2001. doi:10.1090/S0002-9947-01-02730-1.

28 R. P. Stanley. *Combinatorics and commutative algebra*, volume 41 of *Progress in Mathematics*. Birkhäuser Boston, Inc., Boston, MA, second edition, 1996.

29 M. Tancer. Recognition of collapsible complexes is NP-complete. *Discrete Comput. Geom.*, 55(1):21–38, 2016.

30 I.A. Volodin, V.E. Kuznetsov, and A.T. Fomenko. The problem of discriminating algorithmically the standard three-dimensional sphere. *Usp. Mat. Nauk*, 29(5):71–168, 1974. In Russian. English translation: Russ. Math. Surv. 29,5:71–172 (1974).

31 M. L. Wachs. Poset topology: Tools and applications. In *Geometric combinatorics*, volume 13 of *IAS/Park City Math. Ser.*, pages 497–615. American Mathematical Soc., 2007.

32 J. H. C. Whitehead. Simplicial spaces, nuclei and m-groups. *Proc. London Math. Soc. (2)*, 45(1):243–327, 1939. doi:10.1112/plms/s2-45.1.243.

Optimal Morphs of Planar Orthogonal Drawings

Arthur van Goethem
TU Eindhoven
Eindhoven, the Netherlands
a.i.v.goethem@tue.nl

Kevin Verbeek[1]
TU Eindhoven
Eindhoven, the Netherlands
k.a.b.verbeek@tue.nl

Abstract

We describe an algorithm that morphs between two planar orthogonal drawings Γ_I and Γ_O of a connected graph G, while preserving planarity and orthogonality. Necessarily Γ_I and Γ_O share the same combinatorial embedding. Our morph uses a linear number of linear morphs (linear interpolations between two drawings) and preserves linear complexity throughout the process, thereby answering an open question from Biedl et al. [4].

Our algorithm first *unifies* the two drawings to ensure an equal number of (virtual) *bends* on each edge. We then interpret bends as vertices which form obstacles for so-called *wires*: horizontal and vertical lines separating the vertices of Γ_O. We can find corresponding wires in Γ_I that share topological properties with the wires in Γ_O. The structural difference between the two drawings can be captured by the *spirality* of the wires in Γ_I, which guides our morph from Γ_I to Γ_O.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis

Keywords and phrases Homotopy, Morphing, Orthogonal drawing, Spirality

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.42

Related Version A full version of this paper is available at `https://arxiv.org/abs/1801.02455`

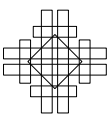
Acknowledgements We want to thank the anonymous reviewers for their extensive feedback.

1 Introduction

A morph is a continuous transformation between two objects. It is most effective, from a visual point of view, if the number of steps during the transformation is small, if no step is overly complex, and if the morphing object retains some similarity to input and output throughout the process. These visual requirements can be translated to a variety of algorithmic requirements that depend on the type of object to be morphed.

In this paper we focus on morphs between two planar orthogonal drawings of a connected graph G with complexity n. In this setting the visual requirements for a good morph can be captured as follows: few (ideally at most linearly many) steps in the morph, each step is a simple (ideally linear) morph, and each intermittent drawing is a planar orthogonal drawing of G with complexity $O(n)$. Biedl et al. [5] presented some of the first results on this topic, for the special case of *parallel* drawings: two graph drawings are parallel when every edge has the same orientation in both drawings. The authors proved that there exists a morph, which

[1] K. Verbeek is supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 639.021.541.



is composed of $O(n)$ linear morphs, between two parallel drawings that maintains parallelity and planarity for orthogonal drawings. More recently, Biedl et al. [4] described a morph, composed of $O(n^2)$ linear morphs, between two planar orthogonal drawings that preserves planarity, orthogonality, and linear complexity during the morph. The authors also present a lower bound example requiring $\Omega(n^2)$ linear morphs when morphed with their method.

In this paper we present a significant improvement upon their work: we describe a morph, which is composed of $O(n)$ linear morphs, between two planar orthogonal drawings which preserves planarity, orthogonality, and linear complexity during the morph. This bound is tight as directly follows from the general lowerbound for straight-line graphs proven in [1].

Related work. Morphs of planar graph drawings have been studied extensively, below we review some of the most relevant results. Cairns showed already in 1944 [7] that there exists a planarity-preserving continuous morph between any two (compatible) triangulations that have the same outer triangle. His proof is constructive and results in an exponential time algorithm to find such a morph. These results were extended in 1983 by Thomassen [12] who showed that two compatible straight-line drawings can be morphed into each other while maintaining planarity (still using exponential time). Thomassen also proved that two rectilinear polygons with the same turn sequence can be transformed into each other using a sequence of linear morphs. Much more recently, Angelini et al. [3] proved that there is a morph between any pair of planar straight-line drawings of the same graph (with the same embedding) using $O(n^2)$ linear morphs. Finally, Alamdari et al. [1] improved this result to $O(n)$ uni-directional linear morphs, which is optimal. Creating this morph takes $O(n^3)$ time. It does create intermediate drawings which need to be represented by a superlogarithmic number of bits, leaving as a final open question if it is possible to morph two planar straight-line drawings using a linear number of linear morphs while using a logarithmic number of bits per coordinate to represent intermediate drawings. Note that, since intermediate drawings are not orthogonal, we cannot apply this approach to our setting. Our approach relies heavily on the spirality of the drawings. This concept of spirality has already received significant attention in the area of bend-minimization (e.g., [6, 8, 9]).

Paper outline. Our input consists of two planar orthogonal drawings Γ_I and Γ_O of a connected graph G, which share the same combinatorial embedding. In Section 2 we first give all necessary definitions and then explain how to create a *unified* graph G: we add "virtual" bends to edges to ensure that each edge is drawn in Γ_I and Γ_O with the same number of bends. We then interpret each bend as a vertex of the unified graph G. Γ_I and Γ_O are now orthogonal straight-line drawings of the unified graph G. Clearly the maximum complexity of Γ_I and Γ_O is still bounded by $O(n)$ after the unification process.

Our main tool are so-called *wires* which are introduced in Section 3. Wires capture the horizontal and vertical order of the vertices. Specifically, we consider a set of horizontal and vertical lines that separate the vertices of Γ_O. If we consider the vertices of Γ_O as obstacles, then these wires define homotopy classes with respect to the vertices of G (for the combinatorial embedding of G shared by Γ_I and Γ_O). These homotopy classes can be represented by orthogonal polylines (also called wires) in Γ_I using orthogonal shortest and lowest paths as defined by Speckmann and Verbeek [11]. A theorem by Freedman, Hass, and Scott [10] proves that the resulting paths minimize crossings.

Intuitively our morph is simply straightening the wires in Γ_I using the *spirality* (the difference between the number of left and right turns) of the wires as a guiding principle. In Section 4 we show how this approach leads more or less directly to a linear number of linear

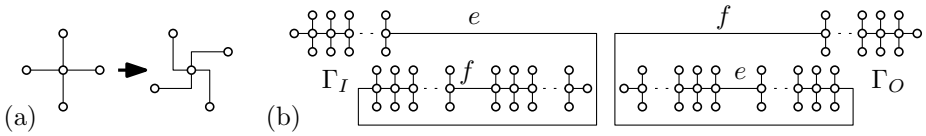


Figure 1 (a) A vertex twist (from [4]). **(b)** The input and output for the lowerbound example from [4]. There are three parts each containing a linear number of vertices. Edge e in Γ_I, respectively f in Γ_O, has a linear number of bends (only four drawn in example).

morphs. However, the complexity of the intermediate drawings created by this algorithm might increase to $\Theta(n^2)$. In Section 5 we show how to refine our approach, to arrive at a linear number of linear morphs which preserve linear complexity of the intermediate drawings.

Relation to Biedl et al. [4]. While the underlying principle of our algorithm is quite different, there are certain similarities between our approach and the one employed by Biedl et al. [4]. As mentioned, there is a lower bound that proves their method cannot do better than $O(n^2)$ linear morphs in general. We sketch why this lower bound does not apply to our algorithm.

To ensure that the spirality of all edges is the same in both the input and the output, the algorithm by Biedl et al. "twists" the vertices (see Fig. 1(a)). The lowerbound described in [4] (see Fig. 1(b)) shows that it may be necessary to twist a linear number of vertices a linear number of times. The complexity introduced in the edges causes another quadratic number of linear moves to keep the complexity of the drawing low.

Our approach must overcome the same problem: a linear number of vertices (edges) might need to be rotated a linear number of times. The crucial difference is that our algorithm can rotate a linear number of vertices (edges) at once, using only $O(1)$ linear morphs. We do not require the edges to have the correct spirality at the start of the morph. Instead we combine the twisting (rotating) of the vertices with linear moves on the edges and pick a suitable order for rotations based on the spirality of the complete drawing. As a result we can change the spirality of a linear number of edges in $\Theta(1)$ linear morphs, and we can rotate a linear number of vertices in $\Theta(1)$ linear morphs. In the full version of the paper we show how our algorithm works on the lowerbound example of [4].

2 Preliminaries

Orthogonal drawings. A *drawing* Γ of a graph $G = (V, E)$ is a mapping of each vertex to a distinct point in the plane and each edge (u, v) to a curve in the plane connecting $\Gamma(u)$ and $\Gamma(v)$. A drawing is *orthogonal* if each curve representing an edge is an orthogonal polyline consisting of horizontal and vertical segments, and a drawing is *planar* if no two curves representing edges intersect in an internal point. Two drawings Γ and Γ' of the same graph G have the same *combinatorial embedding* if at every vertex of G the cyclic order of incident edges is the same in both Γ and Γ'.

Let Γ and Γ' be two planar drawings with the same combinatorial embedding. A *linear morph* Γ_t $(0 \leq t \leq 1)$ from Γ to Γ' consists of a linear interpolation between the two drawings Γ and Γ', that is, $\Gamma_0 = \Gamma$, $\Gamma_1 = \Gamma'$, and for each vertex v, $\Gamma_t(v) = (1-t)\Gamma(v) + t\Gamma'(v)$. A linear morph *maintains planarity* if all intermediate drawings Γ_t are also planar. Note that a linear morph from Γ to Γ' may not maintain planarity even if Γ and Γ' are planar, and that the linear morph may maintain planarity only if Γ and Γ' have the same combinatorial embedding. Therefore, a morph between two planar drawings that maintains planarity generally has to be composed of several linear morphs.

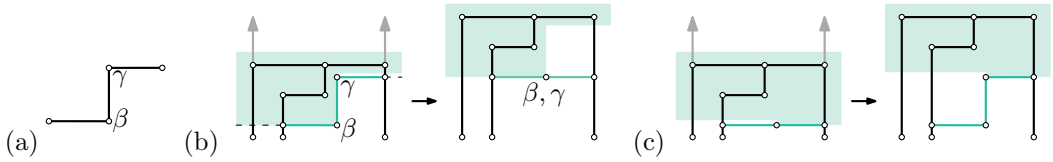


Figure 2 **(a)** A horizontal *zigzag*. **(b)** A *zigzag-eliminating* slide is a linear morph straightening a zigzag. **(c)** A *bend-creating* slide is a linear morph that introduces a zigzag.

Slides. Following Biedl et al. [4] we generally use two types of linear morphs: zigzag-eliminating slides and bend-creating slides. Let a *bend* be the shared endpoint of two consecutive segments of an edge. A *zigzag* consists of three segments joined by two consecutive bends β, γ that form a left followed by a right bend, or vice versa. We call zigzags starting and ending with a horizontal segment *horizontal zigzags* (see Fig. 2(a)), and the rest *vertical zigzags*. We consider the situation with bends β, γ of Fig. 2(a), other situations are symmetric. Let $\mathcal{V}$ be the set of vertices and bends that are strictly left of β and above or at the same height as β, or strictly above γ. Also include β in $\mathcal{V}$. A *zigzag-eliminating slide* moves all points in $\mathcal{V}$ up by the initial distance between β and γ (see Fig. 2(b)). A zigzag-eliminating slide is a linear morph and it straightens the zigzag to a single horizontal or vertical line. The morph always maintains planarity between two drawings.

Inversely, a *bend-creating slide* is a morph that introduces a zigzag in a horizontal or vertical line (see Fig. 2(c)). It can be perceived as the inverse operation of a zigzag-eliminating slide and for similar reasoning is a linear morph that maintains planarity.

Homotopic paths. Our morphing algorithm heavily relies on the concept of *wires* among the vertices of the drawings, and wires are linked up between different drawings via their homotopy classes. We consider the vertices of a drawing as the set of obstacles B. Let $\pi_1, \pi_2 \colon [0,1] \to \mathbb{R}^2 \setminus B$ be two paths in the plane avoiding the vertices. We say that π_1 and π_2 are *homotopic* (notation $\pi_1 \sim_h \pi_2$) if they have the same endpoints and there exists a continuous function avoiding B that deforms π_1 into π_2. More specifically, there exists a function $\Pi : [0,1] \times [0,1] \to \mathbb{R}^2$ such that

- $\Pi(0,t) = \pi_1(t)$ and $\Pi(1,t) = \pi_2(t)$ for all $0 \le t \le 1$.
- $\Pi(s,0) = \pi_1(0) = \pi_2(0)$ and $\Pi(s,1) = \pi_1(1) = \pi_2(1)$ for all $0 \le s \le 1$.
- $\Pi(\lambda,t) \notin B$ for all $0 \le \lambda \le 1$, $0 \le t \le 1$.

Since the homotopic relation is an equivalence relation, every path belongs to a *homotopy class*. The *geometric intersection number* of a pair of paths π_1, π_2 is the minimum number of intersections between any pair of paths homotopic to π_1, respectively π_2. Freedman, Hass, and Scott proved the following theorem[2].

▶ **Theorem 1** (from [10]). *Let M^2 be a closed, Riemannian 2-manifold, and let $\sigma_1 \subset M^2$ and $\sigma_2 \subset M^2$ be two shortest loops of their respective homotopy classes. If $\pi_1 \sim_h \sigma_1$ and $\pi_2 \sim_h \sigma_2$, then the number of crossings between σ_1 and σ_2 is at most the number of crossings between π_1 and π_2.*

In other words, the number of crossings between two loops of fixed homotopy classes are minimized by the shortest respective loops. This theorem can easily be extended to paths instead of loops, if we can consider the endpoints of the paths as obstacles. For orthogonal

[2] Reformulated (and simplified) to suit our notation rather than the more involved notation in [10].

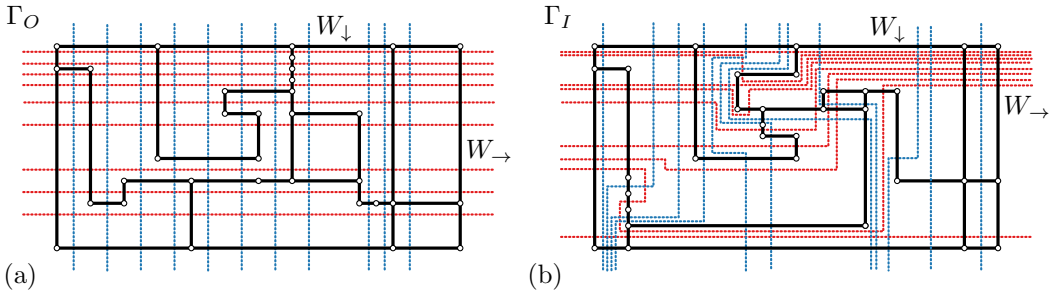


Figure 3 **(a)** The set $W_\rightarrow$ of lr-wires (red) and the set $W_\downarrow$ of tb-wires (blue) in the output drawing Γ_O. **(b)** The matching wires in the input drawing Γ_I.

paths, the shortest path is not uniquely defined and the theorem cannot directly be applied. However, using *lowest paths* the theorem still holds. Refer to [11] (Lemma 6) for details.

Conventions. When morphing from a drawing Γ to a drawing Γ', the complexities (number of vertices and bends) of the two drawings may not be the same, as there is no restriction on the complexity of the orthogonal polylines representing the edges. To simplify the discussion of our algorithm, we first ensure that every two orthogonal polylines in Γ and Γ' representing the same edge have the same number of segments. This can easily be achieved by subdividing segments, creating additional virtual bends. Next, we eliminate all bends by replacing them with vertices. As a result, all edges of the graph are represented by straight segments (horizontal or vertical) in both Γ and Γ', and there are no bends. We call the resulting graph the *unification* of Γ and Γ'. If the maximal complexity of Γ and Γ' is $O(n)$ then clearly the complexity of the unification of Γ and Γ' is $O(n)$.

We say that two planar drawings Γ and Γ' of a unified graph are *similar* if the horizontal and vertical order of the vertices is the same in both drawings. A planar drawing can be morphed to a similar planar drawing using a single linear morph while maintaining planarity. We can only introduce a crossing if two vertices swap order in the horizontal or vertical direction, which cannot happen during a linear morph between two similar drawings.

Finally, when morphing between two planar drawings Γ and Γ' of a graph G, we assume that Γ and Γ' have the same combinatorial embedding and the same outer boundary. Furthermore, we assume that G is connected. If G is not connected, then we can use the result by Aloupis et al. [2] to connect G in a way that is compatible with both Γ and Γ'. Doing so might increase the complexities of the drawings to $O(n^{1.5})$.

3 Wires

In the following we assume that we want to morph an orthogonal planar drawing Γ_I of $G = (V, E)$ to another orthogonal planar drawing Γ_O of G while maintaining planarity and orthogonality. We further assume that Γ_I and Γ_O have the same combinatorial embedding and the same outer boundary. We also assume that G is connected, G is the unification of Γ_I and Γ_O, and that G contains n vertices.

To morph Γ_I to Γ_O, our main strategy is to first make Γ_I similar to Γ_O, after which we can morph Γ_I to Γ_O using a single linear morph. To capture the horizontal and vertical order of the vertices, we use two sets of *wires*. The *lr-wires* $W_\rightarrow$, going from left to right through the drawings, capture the vertical order of the vertices. The *tb-wires* $W_\downarrow$, going from top to bottom through the drawings, capture the horizontal order of the vertices.

Since we want to match the horizontal and vertical order of vertices in Γ_O, the wires $W_\rightarrow$ and $W_\downarrow$ are simply horizontal and vertical lines in Γ_O, respectively, separating any two consecutive coordinates used by vertices (see Fig. 3(a)). Now assume that we have a planar morph from Γ_I to Γ_O (the existence of such a morph follows from [4]). If we were to apply this morph in the reverse direction on the wires in Γ_O, we end up with another set of wires in Γ_I with the following properties (see Fig. 3(b)): (1) the order of the wires in $W_\rightarrow$ ($W_\downarrow$) is the same as in Γ_O and the same vertices are between consecutive wires, (2) two wires are non-crossing if they both belong to $W_\rightarrow$ or $W_\downarrow$ and cross exactly once otherwise, and (3) the wires cross exactly the same sequence of edges as in Γ_O. These properties follow directly from the fact that a planar morph cannot introduce or remove any crossings, and thus these properties are invariant under planar morphs. We say that a set of wires is *proper* if it has the above properties. Interestingly, any proper set of wires can be used to construct a planar morph from Γ_I to Γ_O. We first use a planar morph to straighten the wires. Then, by Property (1), the resulting drawing Γ is similar to Γ_O, except that it may have some extra bends. However, by Property (2) the wires form a grid where each cell contains at most one vertex, and the edges crossing the wires are correct by Property (3). Hence, we can eliminate all bends in a single morph by combining individual morphs per cell. For each cell we morph all bends (and the vertex) to the center of the cell. The resulting drawing is similar to Γ_O and has no bends, and thus we can finish the planar morph with a single linear morph.

In the following we assume that we are given a proper set of wires in Γ_I. Our first goal is to straighten these wires. To keep the distinction between wires and edges clear, we refer to the horizontal and vertical segments of wires as *links*. Note that even a single wire in Γ_I may have $\Omega(n^2)$ links (see Fig. 4), so it is not efficient to straighten the wires one link at a time. To straighten the wires more efficiently, we consider the spirality of the wires. For a wire $w \in W_\rightarrow$, let $\ell_1 \dots \ell_k$ be the links of w in order from left to right. Furthermore, let b_i be the orientation of the bend between ℓ_i and ℓ_{i+1}, where $b_i = 1$ for a left turn, $b_i = -1$ for a right turn, and $b_i = 0$ otherwise. The *spirality* of a link ℓ_i is defined as $s(\ell_i) = \sum_{j=1}^{i-1} b_j$. Note that, by definition, the spirality of ℓ_1 is 0, and by construction the spirality of ℓ_k is also 0. The *spirality* of a wire is defined as the maximum absolute value of the spirality over all its links. The spirality of wires in $W_\downarrow$ is defined analogously, going from top to bottom.

▶ **Lemma 2.** *If a wire $w \in W_\rightarrow$ and a wire $w' \in W_\downarrow$ cross in links ℓ_i and ℓ'_j, then $s(\ell_i) = s(\ell'_j)$.*

Proof. By Property (2) w and w' cross exactly once. Consider an axis-aligned rectangle R that contains the complete drawing and that intersects the first and last link of both w and w'. By definition the spirality of w and w' is zero where they intersect R. The wires w and w' subdivide R into four simple faces (see Fig. 5). Consider the top-left face. Since the face is simple, a counterclockwise tour of the face would increase spirality by four. As R and the intersection of R with w and w' contribute three left turns, the spirality should increase by one when traversing the face from the first link of w to the first link of w', where the spiralities of ℓ_1 and ℓ'_1 are 0. Assuming that the spirality of ℓ_i is x and the spirality of ℓ'_j is y, then we get that $x + 1 - y = 1$ (including the left turn at the crossing). Thus $x = y$. ◀

Spirality bound. In Section 4 we show that we can straighten a set of wires using only $O(k)$ linear morphs, if the spirality of each wire is bounded by k. It is therefore pertinent to bound the spirality of a proper set of wires. For that we use a particular set of wires. First consider any proper set of wires, which must exist. Then replace every wire w by the lowest and shortest path homotopic to w. Because the new wires are homotopic to the initial wires, Property (1) is maintained. Although the wires may partially overlap, they cannot properly

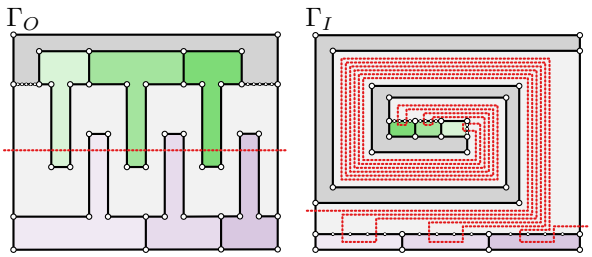


Figure 4 The complexity of a wire can be $\Omega(n^2)$. To satisfy Property (3), the wire in Γ_I must spiral through the same polygon a linear number of times. Note that the spirality is still $O(n)$.

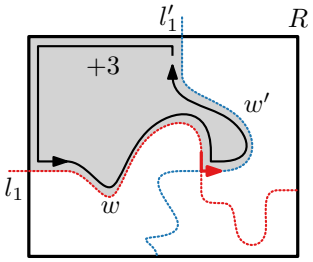


Figure 5 As link l_1 and l'_1 both have spirality zero and a counterclockwise tour increases spirality by four, the crossing links of w and w' must have equal spirality.

cross, and thus overlaps can be removed by slight perturbations. Furthermore, Properties (2) and (3) follow directly from Theorem 1 and Lemma 6 from [11]. Thus, the new set of wires is proper, and in the following we can assume that all wires are lowest and shortest.

We show that the spirality of any wire in Γ_I is $O(n)$. We first establish some properties of the spiralities of links. We consider wires in $W_\rightarrow$ and links with positive spirality, but analogous or symmetric results hold for wires in $W_\downarrow$ and links with negative spirality.

▶ **Lemma 3.** *Let ℓ_i be a horizontal link of a wire $w \in W_\rightarrow$ with even spirality $s(\ell_i) \geq 2$ and let L be a vertical line crossing ℓ_i. Then there exists a link ℓ_j $(j < i)$ of w with spirality $s(\ell_i) - 2$ or $s(\ell_i) - 4$ that crosses L below ℓ_i.*

Proof. Let $w[i]$ be the partial wire consisting of links $\ell_1 \dots \ell_i$ of w. We first argue that if ℓ_i is the lowest link of $w[i]$ crossing L, then $s(\ell_i) \leq 0$. In this case we can create a simple cycle in counterclockwise direction by (1) first going down from ℓ_1 (starting sufficiently far to the left and going down far enough to avoid crossing the rest of $w[i]$), (2) going to the right until reaching L, (3) going up until reaching ℓ_i, and (4) following $w[i]$ backwards until reaching the starting point of the cycle. The full cycle should be of spirality 4, and it already contains 3 left turns by construction. Let x be the contribution of the turn at ℓ_i (which can be left or right). Then we get that $3 + x - s(\ell_i) = 4$ or $s(\ell_i) = x - 1$, which directly implies $s(\ell_i) \leq 0$. Let $\ell_k \in w[i]$ be the link crossing L directly below ℓ_i (with $k < i$). We can again construct a simple cycle consisting of the wire $w[i]$ from ℓ_k to ℓ_i and the segment of L connecting ℓ_k and ℓ_i (see Fig 6(a)). This cycle contains the bends between ℓ_k and ℓ_i on $w[i]$, and two bends at the crossings with L. Following the cycle in counterclockwise direction implies that there cannot be right turns at both the crossing between L and ℓ_i and between L and ℓ_k, for otherwise $w[k]$ would have to cross L between ℓ_k and ℓ_i (see Fig. 6(b)), contradicting our assumption. Therefore, the bends of $w[i]$ between ℓ_k and ℓ_i contribute between 2 and 4 to the total spirality of 4 of the cycle (in either direction). As a result, the spirality between ℓ_k and ℓ_i can differ by at most 4. We can repeat this argument on $w[k]$. Since the lowest link crossing L has spirality at most 0, there must exist a link ℓ_j crossing L below ℓ_i with spirality $s(\ell_i) - 2$ or $s(\ell_i) - 4$. ◀

If we consider a link ℓ_i with negative spirality, then Lemma 3 holds for a link ℓ_j with spirality $s(\ell_i) + 2$ or $s(\ell_i) + 4$ that intersects L above ℓ_i. By repeatedly applying Lemma 3 we obtain the following result.

▶ **Lemma 4.** *For each link of a wire $w \in W_\rightarrow$ with positive (negative) spirality s there exists a vertical line L and a subsequence of $\Omega(|s|)$ links of w crossing L from bottom to top (top to bottom), such that these links are also ordered increasingly (decreasingly) on spirality.*

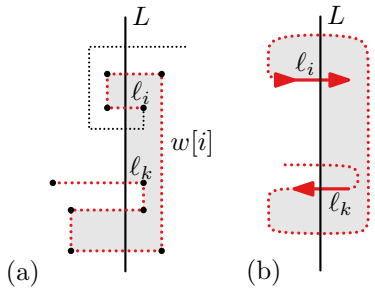


Figure 6 (a) The sub-wire $w[i]$ (red). The path along $w[i]$ from ℓ_k to ℓ_i and the segment of L connecting ℓ_i and ℓ_k forms a simple cycle. **(b)** The cycle cannot have two right turns adjacent to L as $w[i]$ does not intersect L between ℓ_k and ℓ_i.

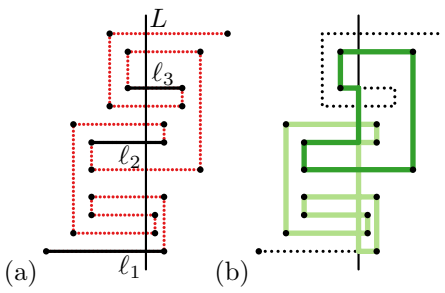


Figure 7 (a) A wire of spirality $s = 4$ and $\Omega(s)$ selected edges that are ordered both along $w[i]$ and L. **(b)** A decomposition of w in two polygonal lines and $\Omega(s)$ non-overlapping cycles.

▶ **Lemma 5.** *The maximum absolute spirality of any link of a wire $w \in W_\rightarrow$ in Γ_I is $O(n)$.*

Proof. Without loss of generality assume that the maximum absolute spirality of a link in wire w occurs for a link with positive spirality, and let that spirality be s. By Lemma 4 there exists a subsequence $\ell_1, \ldots, \ell_k$ of links of w (not necessarily consecutive along w) ordered increasingly by spirality with $k = \Omega(s)$, such that all of these links cross a vertical line L in order from bottom to top. We can thus construct $k - 1$ simple cycles by connecting ℓ_i to ℓ_{i+1} along L and along w, such that two different cycles do not share any segments (see Fig. 7). Since w is constructed to be shortest with respect to its homotopy class, and G is connected, every cycle constructed in this way must cross an edge of Γ_I, for otherwise w can be shortened by following L locally. By construction L can only cross $O(n)$ edges of Γ_I, and each edge only once. Similarly, w can cross only $O(n)$ edges of Γ_O, and each edge only once, as w is horizontal in Γ_O. As the wires are proper w also crosses only $O(n)$ edges in Γ_I. Therefore, the cycles can cross only $O(n)$ edges in total, and thus $s = O(k) = O(n)$. ◀

Analogously, Lemma 5 also holds for wires in $W_\downarrow$, and thus the maximum spirality of all wires is bounded by $O(n)$. Note that we only use shortest paths to bound the spirality of the wires. In the remainder of the paper we merely require that the spirality is bounded by $O(n)$, and any proper set of wires satisfying that bound suffices for our purpose.

4 Linear number of linear morphs

We now describe our algorithm to morph Γ_I to Γ_O using $O(n)$ linear morphs. The complexity of the drawing may grow to $O(n^2)$ intermediately though. In Section 5 we refine our approach to keep the complexity of intermediate drawings at $O(n)$.

It is important to note that for our analysis of the initial spirality we required Γ_I and Γ_O to be straight-line drawings of the unified graph. For the morph itself we let go of this stringent requirement. During the morph we introduce bends in the edges to rotate them. We will show that the spirality of the wires only decreases during the morph.

The idea of the algorithm is to reduce the maximum spirality of the wires using only $O(1)$ linear morphs. Then, by Lemma 5 we need only $O(n)$ linear morphs to straighten the wires, after which we can morph the drawing to Γ_O using $O(1)$ linear morphs, as described in Section 3. We will show that we can reduce the spirality of wires in $W_\rightarrow$ ($W_\downarrow$) without increasing the spirality of wires in $W_\downarrow$ ($W_\rightarrow$) and vice versa. In the description below, we limit ourselves to straightening the wires in $W_\rightarrow$.

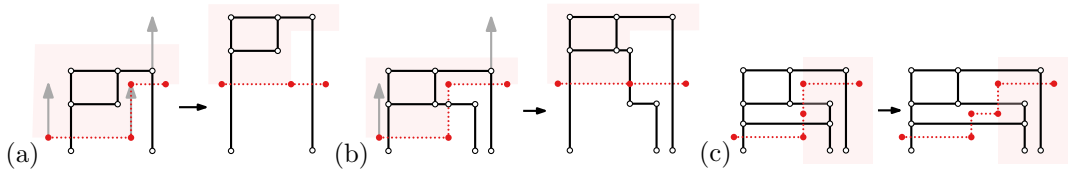


Figure 8 Different slide-types used on the wires. **(a)** The slide from [4] executed on a wire. **(b)** A single crossing edge (link) causes the introduction of new bends in the edge (link). **(c)** A bend-introducing slide offsets edge intersections without increasing the spirality of the wire.

Now let ℓ^* be a link with maximum absolute spirality. To reduce the absolute spirality of ℓ^*, we use a zigzag-eliminating slide as described in Section 2, where ℓ^* is the middle link of the zigzag. As ℓ^* is a link with maximum absolute spirality, the links adjacent to ℓ^* are on opposite sides of the line through ℓ^*. It is easy to see that this slide thus eliminates ℓ^* and does not introduce any bends in the wires in $W_\rightarrow$ (see Fig. 8(a)). However, the link ℓ^* may intersect an edge of Γ_I or a link of a wire from $W_\downarrow$. In that case we introduce a bend in the involved edge (link) to execute the slide properly (see Fig. 8(b)). If ℓ^* intersects more than one edge of Γ_I, then we must be careful not to introduce an overlap in Γ_I. To avoid this, we first execute bend-creating slides, essentially subdividing ℓ^*, to ensure that every link with maximum absolute spirality intersects with at most one edge of Γ_I (see Fig. 8(c)).

To reduce the number of linear morphs, we combine all slides of the same type into a single linear morph. For all links with the same spirality, all bend-creating slides are combined into one linear morph, and all zigzag-eliminating slides are combined into another linear morph. Links with positive spirality and links with negative spirality are combined into separate linear morphs. Thus, using at most 4 linear morphs, we reduce the maximum spirality of all wires in $W_\rightarrow$ by one.

Analysis. We first show that performing slides on links in $W_\rightarrow$ does not have adverse effects on wires in $W_\downarrow$. This is easy to see for bend-creating slides, as we can assume that wires in $W_\rightarrow$ and wires in $W_\downarrow$ never have overlapping links.

▶ **Lemma 6.** *Performing a zigzag-eliminating slide on a link with maximum absolute spirality in $W_\rightarrow$ does not increase the spirality of a wire in $W_\downarrow$.*

Proof. Let ℓ^* be the middle link of the zigzag-eliminating slide. The zigzag-eliminating slide can only change a wire w' in $W_\downarrow$ if ℓ^* crosses a link ℓ' in w'. By Lemma 2 $s(\ell') = s(\ell^*)$. The slide does not change the spirality of any link in w', but a new link has been introduced in the middle of ℓ'. This new link in w' crosses the link obtained by eliminating ℓ^*, which has absolute spirality $|s(\ell^*)| - 1$. By Lemma 2 the new link in w' must also have absolute spirality $|s(\ell^*)| - 1$, and thus the spirality of w' has not been increased. ◀

We also prove that we can combine zigzag-eliminating slides (and bend-creating slides) into a single linear morph that maintains both planarity and orthogonality of the drawing.

▶ **Lemma 7.** *Multiple bend-creating or zigzag-eliminating slides on links of the same spirality in $W_\rightarrow$ can be combined into a single linear morph that maintains planarity and orthogonality.*

Proof. As bend-creating slides are simply the inverse of zigzag-eliminating slides, we can restrict ourselves to the latter. As all zigzag-eliminating slides operate on links of the same spirality, they are all either horizontal or vertical. Without loss of generality, assume that all zigzags are horizontal. Then all vertices in the drawing are moved only vertically, which

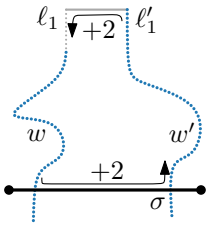


Figure 9 Two wires crossing the same segment σ have the same spirality as $s(\ell_1) = s(\ell'_1) = 0$ and a counterclockwise tour increases spirality by four.

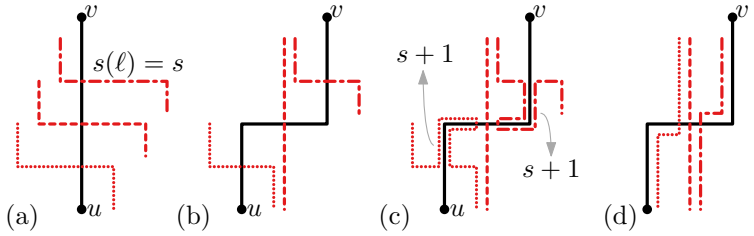


Figure 10 **(a)** A segment $\sigma = (u, v)$ with three crossing wires of maximum spirality $s > 0$. **(b)** A slide on an arbitrary crossing link adds two bends to σ. **(c)** Rerouting the remaining wires ensures all crossing links have spirality $s - 1$. New links are created with spirality $s + 1$ (and of sprirality s) but they do not cross any edge. **(d)** Result after reducing all non-crossing links of spirality s and $s + 1$.

means that the horizontal order of vertices is maintained and that vertical edges remain vertical. Furthermore, since we introduce bends at edges that intersect the middle segment of zigzags, horizontal edges are either subdivided or remain horizontal during the linear morph. Finally, we can only violate planarity if a vertex overtakes an edge in the vertical direction. However, by construction, points with higher y-coordinates are moved up at least as far as points with lower y-coordinates, and thus the vertical order is also maintained. ◀

▶ **Theorem 8.** *Let Γ_I and Γ_O be two orthogonal planar drawings of G, where G is the unification of Γ_I and Γ_O, and Γ_I and Γ_O have the same combinatorial embedding and the same outer boundary. Then we can morph Γ_I to Γ_O using $O(n)$ linear morphs, where n is the number of vertices of G.*

Proof. Let $W_\rightarrow$ and $W_\downarrow$ be a proper set of wires for Γ_I with maximum spirality $O(n)$. As shown in Section 3 such a set exists. Using Lemma 7, we repeatedly reduce the maximum spirality of the wires in $W_\rightarrow$ and $W_\downarrow$ by one using at most two times 4 linear morphs as described above. By Lemmata 5 and 6 all wires can be straightened with at most $O(n)$ linear morphs. Afterwards, the resulting drawing Γ is similar to Γ_O except for additional bends. Using $O(1)$ linear morphs we can morph Γ to Γ_O (see Section 3). ◀

5 Linear complexity

We refine the approach from Section 4 to ensure that the drawing maintains $O(n)$ complexity during the morph. To achieve this we make two small changes to the algorithm. First, we ensure that for each edge intersected by links of maximum absolute spirality we only perform a slide for one of the intersecting links and reroute the remaining wires. This ensures that only $O(1)$ bends per edge are added per iteration of the algorithm. Second, we perform additional intermittent linear morphs to keep the number of bends per edge low. Both alterations add only $O(n)$ additional linear morphs in total. The changes ensure that each edge has $O(1)$ bends at every step of the morph; the $O(n)$ complexity bound trivially follows.

Rerouting wires. During each iteration of the algorithm in Section 4 we add $O(1)$ linear morphs to ensure that only $O(1)$ new bends are introduced in each edge. Our approach maintains the invariant that all wires crossing an edge cross the same segment of the edge. Trivially this is the case in Γ_I. We first establish the following property.

▶ **Lemma 9.** *All links intersecting the same segment of an edge have the same spirality.*

Proof. Each edge e has a unique orientation in Γ_O, hence either only wires from $W_\rightarrow$ or $W_\downarrow$ intersect e. All wires cross e in the same direction. Assume without loss of generality that e is horizontal in Γ_O and thus only intersected by wires from $W_\downarrow$. Consider two adjacent wires $w, w' \in W_\downarrow$ intersecting segment σ of edge e. Consider an additional horizontal segment that would connect link ℓ_1 of w and ℓ'_1 of w' (if needed extend ℓ_1 or ℓ'_1 upwards). The area enclosed by wires w, w', the segment σ, and the extra horizontal segment forms a simple cycle (see Fig. 9). In this cycle there are two left turns at σ and two left turns at the additional horizontal segment, and the remaining bends belong to w and w'. If x and x' are the spiralities of w and w' when intersecting σ, then $x + 2 - x' + 2 = 4$, and thus $x = x'$. ◀

If multiple links cross a single edge, we execute a slide on only one of these links. We then reroute the remaining wires crossing the edge. This may introduce links with higher absolute spirality, but we can eliminate these links using $O(1)$ linear morphs without affecting the complexity of the drawing. This is formalized in the following lemma.

▶ **Lemma 10.** *The maximum absolute spirality of all links can be reduced while increasing the complexity of each edge by at most $O(1)$.*

Proof. Let s be the spirality with the maximum absolute value. For each edge e crossed by multiple links with spirality s, perform a slide for a single crossing link. By our invariant all links cross e in the same segment σ (see Fig. 10(a)). Performing a slide on an arbitrary crossing link introduces two new bends in e (see Fig. 10(b)). We now reroute the remaining crossing wires in an ε band along the edge to cross e in the newly created segment (see Fig. 10(c)). As for a small enough ε no edge or other wire will be in the area of rerouting, the wires remain a proper set. The absolute spirality of the crossing links is now $|s| - 1$. The remaining newly created links have absolute spirality at most $|s| + 1$.

By Lemma 7 $O(1)$ linear morphs are sufficient to perform a slide on all selected crossing links. Similarly $O(1)$ linear morphs are sufficient to remove all links of absolute spirality $|s| + 1$ and then all links of absolute spirality $|s|$ (see Fig. 10(d)). As none of the latter links intersect an edge this does not affect the complexity of the drawing. ◀

Removing excess bends. Rerouting wires ensures that every edge gathers only $O(1)$ bends when reducing the spirality of all intersecting links by one. However as the maximum absolute spirality, as well as the complexity of Γ_I, is $O(n)$, the total complexity of the drawing may still become $O(n^2)$ during the morph. We show that using an additional $O(1)$ linear morphs per iteration we can also maintain $O(n)$ complexity.

▶ **Lemma 11.** *At any point during the morph, the bends with left orientation in an edge are separated from the bends with right orientation by the wires crossing the edge.*

Proof. Consider an arbitrary wire w crossing edge $e = (u, v)$. Let ℓ be the link of w that crosses e and assume without loss of generality that $s(\ell) > 0$ and that u is on the left side of w. Let σ be the segment of e crossed by ℓ. We show that when traversing e all right oriented bends occur before the crossing with w and all left oriented bends occur after. The claim trivially follows. We consider the orientation of the bends when traversing e from u to v.

Clearly the claim already holds in Γ_I. Now consider a drawing Γ during the morph, where ℓ has maximum absolute spirality, and assume the property holds in Γ. As $s(\ell) > 0$, ℓ must be preceded by a left turn and followed by a right turn. Performing a zigzag-eliminating slide on ℓ will merge these links into a new link ℓ'. A right bend u' is introduced in σ left of the intersection with ℓ', and thus on the side of u, and a left bend v' right of the intersection.

But then when traversing e all right-oriented bends in e occur before the crossing with w and all left-oriented bends occur after. ◀

We define a *cell* as the area enclosed by two consecutive wires in $W_\rightarrow$ and two consecutive wires in $W_\downarrow$. By the properties of a proper set of wires, each cell can contain at most one vertex and each edge incident to such a vertex must intersect a different wire. We now use the following simple approach. Let Γ be the drawing after an arbitrary iteration of the algorithm. If a cell in Γ contains at least two bends on each edge incident to the vertex of that cell, then we perform $O(1)$ linear morphs to eliminate a bend on each of the incident edges. We can combine the linear morphs for all separate cells.

▶ **Lemma 12.** *At any point during the morph, inside a single cell for any pair of edges the number of bends differs by at most a constant.*

Proof. The statement is vacuously true for cells without a vertex, so consider an arbitrary vertex v and the cell it is contained in. To prove the statement for Γ we first consider the spirality of the links intersecting incident edges of v in Γ_I. We show that in Γ_I for two incident edges e_1, e_2 that are adjacent in the cyclic order at v the spirality of the links crossing e_1 differs by at most 2 from those crossing e_2. During the morph we always perform slides on links with maximum absolute spirality and introduce exactly 1 bend inside a cell to reduce the spirality of all links intersecting an edge. It follows that at any time the difference in the number of bends in incident edges inside the cell is bounded by a constant.

Edges e_1 and e_2 have two different possible configurations in Γ_O. Either one is vertical and the other horizontal, or both are horizontal (vertical). We consider the case where one is horizontal and the other vertical. Without loss of generality consider that e_1 and e_2 are above, respectively, to the left of v in Γ_O. By construction e_1 and e_2 are intersected by a pair of wires $w \in W_\rightarrow$ and $w' \in W_\downarrow$, and they cross before crossing e_1 respectively e_2. Wires w and w' together with edges e_1 and e_2 must then enclose a simple cycle in Γ_O. As the wires form a proper set this must also be the case in Γ_I, however, the orientation of the edges may be different in Γ_I. The cycle contains three left-corners by construction (see Fig. 11(a)). The turn at v depends on the configuration of e_1 and e_2 in Γ_I. Let ℓ, ℓ' be the links of w, w' crossing e_1 and e_2 and let k be the spirality of the links at the crossing of w and w'. We have that $(k - s(\ell)) + (s(\ell') - k) + 3 + c = 4$ for $-1 \leq c \leq 1$, and thus $|s(\ell') - s(\ell)| \leq 2$.

For the case where both edges are horizontal (vertical) in Γ_O a similar argument holds, but now the cycle is formed by two wires from $W_\rightarrow$ and one wire from $W_\downarrow$ resulting in one more left turn. We obtain $s(\ell') - s(\ell) + 4 + c = 4$ for $-1 \leq c \leq 1$, resulting in $|s(\ell') - s(\ell)| \leq 1$. ◀

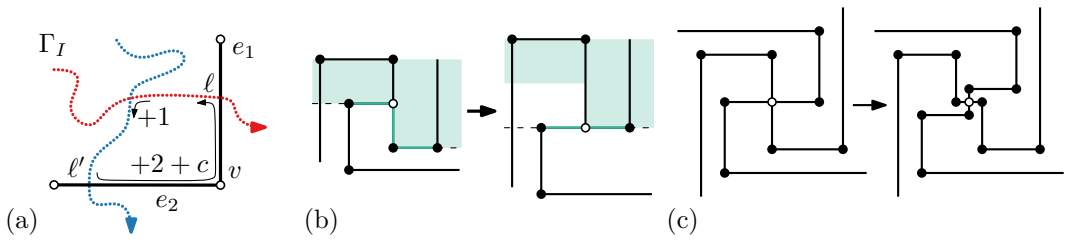


Figure 11 **(a)** The difference of the spirality of links ℓ and ℓ' in Γ_I is at most two as a counterclockwise tour increases spirality by four. The value of c depends on the actual configuration of e_1 and e_2 and ranges between -1 and 1. **(b)** A vertex of degree at most three where all incident edges have at least two bends can be simplified through a series of $O(1)$ zigzag-removing slides. **(c)** The edges incident to a vertex of degree four can be offset an epsilon amount, after which zigzag-removing slides can reduce the number of bends.

▶ **Corollary 13.** *All links crossing incident edges of a degree four vertex have the same spirality in* Γ_I*.*

Proof. For a vertex of degree four two incident edges adjacent in the cyclic order have a fixed relative configuration. We require $c = 1$, resulting in $s(\ell) = s(\ell')$. ◀

▶ **Lemma 14.** *If all edges incident to a vertex* v *have at least two bends inside the same cell, then one bend can be removed from each edge without affecting the rest of the drawing.*

Proof. If v has at most three incident edges, then there always exists a series of zigzag-removing slides that, in a cyclic order, removes one bend from each of the incident edges without affecting the other edges and without losing planarity (see Fig. 11(b)). So assume that v has four incident edges.

By Corollary 13 the spirality of all intersecting links of incident edges is the same in Γ_I. Specifically this the spirality is either positive or negative for all intersecting links. Using Lemma 11 this implies that all edges will either form only left turns or only right turns inside this cell during the morph. Assume without loss of generality that all incident edges have only left turns inside this cell and each has at least two left turns. We simultaneously offset all segments incident to v by an epsilon amount, creating a right bend near v in each edge (see Fig. 11(c)). As we only move the segments an epsilon amount we can safely do so without causing new intersections. Now every incident edge starts with a right-bend followed by a left-bend. Using zigzag-removing slides we remove the newly introduced bend and one of the left-bends. As these zigzags do not intersect any edge or wire this does not change the spirality of any wire or increase the complexity of any edge. We can merge the different moves for all vertices together into $O(1)$ linear morphs. ◀

As each iteration of the refined algorithm increases the complexity of each edge by at most 2 bends, it is sufficient to reduce complexity of the edges once per iteration. By Lemma 14 we can simultaneously simplify all cells where all edges have at least two bends using $O(1)$ linear morphs. And by Lemma 12 this requirement is already met when cells contain only $O(1)$ bends. Cells that do not contain a vertex also do not contain bends as all wires intersect in the same segment of an edge. It directly follows that the complexity of the drawing is $O(n)$ at all times. Furthermore, we still need only a linear number of linear morphs.

▶ **Theorem 15.** *Let* Γ_I *and* Γ_O *be two orthogonal planar drawings of* G*, where* G *is the unification of* Γ_I *and* Γ_O*, and* Γ_I *and* Γ_O *have the same combinatorial embedding and the same outer boundary. Then we can morph* Γ_I *to* Γ_O *using* $O(n)$ *linear morphs while maintaining* $O(n)$ *complexity during the morph, where* n *is the number of vertices of* G*.*

6 Conclusion

We described an algorithm that morphs between two planar orthogonal drawings of a connected graph G using only $O(n)$ linear morphs while maintaining planarity and linear complexity of the drawing during the complete morph. This answers the open question from Biedl et al. [4]. As $\Omega(n)$ linear morphs are needed in the worst case, our algorithm is optimal for connected graphs.

Our current proofs only hold for connected graphs. Specifically Lemma 5 assumes that the graph is connected to argue that each cycle must intersect an edge. By combining the results of Aloupis et al. [2] with our work we also obtain an algorithm requiring only $O(n^{1.5})$ linear morphs for disconnected graphs, which still improves on the $O(n^2)$ bound of [4]. For future work we will investigate if the proofs can be changed to include disconnected graphs.

References

1. Soroush Alamdari, Patrizio Angelini, Fidel Barrera-Cruz, Timothy Chan, Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, Penny Haxell, Anna Lubiw, Maurizio Patrignani, Vincenzo Roselli, Sahil Singla, and Bryan Wilkinson. How to morph planar graph drawings. *SIAM Journal on Computing*, 46(2):824–852, 2017.
2. Greg Aloupis, Luis Barba, Paz Carmi, Vida Dujmović, Fabrizio Frati, and Pat Morin. Compatible connectivity augmentation of planar disconnected graphs. *Discrete & Computational Geometry*, 54(2):459–480, 2015.
3. Patrizio Angelini, Fabrizio Frati, Maurizio Patrignani, and Vincenzo Roselli. Morphing planar graph drawings efficiently. In *Proc. 21st International Symposium on Graph Drawing*, pages 49–60, 2013.
4. Therese Biedl, Anna Lubiw, Mark Petrick, and Michael Spriggs. Morphing orthogonal planar graph drawings. *ACM Transactions on Algorithms*, 9(4):29:1–29:24, 2013.
5. Therese Biedl, Anna Lubiw, and Michael Spriggs. Morphing planar graphs while preserving edge directions. In *Proc. 13th International Symposium on Graph Drawing*, pages 13–24, 2006.
6. Thomas Bläsius, Sebastian Lehmann, and Ignaz Rutter. Orthogonal graph drawing with inflexible edges. *Computational Geometry*, 55:26–40, 2016.
7. Steward Cairns. Deformations of plane rectilinear complexes. *The American Mathematical Monthly*, 51(5):247–252, 1944.
8. Giuseppe Di Battista, Giuseppe Liotta, and Francesco Vargiu. Spirality and optimal orthogonal drawings. *SIAM Journal on Computing*, 27(6):1764–1811, 1998.
9. Walter Didimo, Francesco Giordano, and Giuseppe Liotta. Upward spirality and upward planarity testing. *SIAM Journal on Discrete Mathematics*, 23(4):1842–1899, 2009.
10. Michael Freedman, Joel Hass, and Peter Scott. Closed geodesics on surfaces. *Bulletin of the London Mathematical Society*, 14(5):385–391, 1982.
11. Bettina Speckmann and Kevin Verbeek. Homotopic c-oriented routing with few links and thick edges. *Computational Geometry*, 67:11–28, 2018.
12. Carsten Thomassen. Deformations of plane graphs. *Journal of Combinatorial Theory, Series B*, 34(3):244–257, 1983.

Computational Topology and the Unique Games Conjecture

Joshua A. Grochow[1]
Department of Computer Science & Department of Mathematics
University of Colorado, Boulder, CO, USA
jgrochow@colorado.edu
https://orcid.org/0000-0002-6466-0476

Jamie Tucker-Foltz[2]
Amherst College, Amherst, MA, USA
jtuckerfoltz19@amherst.edu

Abstract

Covering spaces of graphs have long been useful for studying expanders (as "graph lifts") and unique games (as the "label-extended graph"). In this paper we advocate for the thesis that there is a much deeper relationship between computational topology and the Unique Games Conjecture. Our starting point is Linial's 2005 observation that the only known problems whose inapproximability is equivalent to the Unique Games Conjecture – Unique Games and Max-2Lin – are instances of Maximum Section of a Covering Space on graphs. We then observe that the reduction between these two problems (Khot–Kindler–Mossel–O'Donnell, FOCS '04; SICOMP '07) gives a well-defined map of covering spaces. We further prove that inapproximability for Maximum Section of a Covering Space on (cell decompositions of) closed 2-manifolds is also equivalent to the Unique Games Conjecture. This gives the first new "Unique Games-complete" problem in over a decade.

Our results partially settle an open question of Chen and Freedman (SODA, 2010; Disc. Comput. Geom., 2011) from computational topology, by showing that their question is almost equivalent to the Unique Games Conjecture. (The main difference is that they ask for inapproximability over $\mathbb{Z}_2$, and we show Unique Games-completeness over $\mathbb{Z}_k$ for large k.) This equivalence comes from the fact that when the structure group G of the covering space is Abelian – or more generally for principal G-bundles – Maximum Section of a G-Covering Space is the same as the well-studied problem of 1-Homology Localization.

Although our most technically demanding result is an application of Unique Games to computational topology, we hope that our observations on the topological nature of the Unique Games Conjecture will lead to applications of algebraic topology to the Unique Games Conjecture in the future.

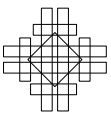
2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness, Theory of computation → Computational geometry

Keywords and phrases Unique Games Conjecture, homology localization, inapproximability, computational topology, graph lift, covering graph, permutation voltage graph, cellular graph embedding

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.43

Related Version A full version of this paper is available as arXiv:1803.06800 [cs.CC].

[1] J. A. G. gratefully acknowledges support from NSF grants DMS-1750319 and DMS-1622390 during the course of this work.
[2] J. T.-F. gratefully acknowledges the support of his Schupf Scholarship at Amherst College.



Acknowledgements We thank Alex Kolla for a great talk on Unique Games that inspired us in this direction. We thank Ryan O'Donnell, Hsien-Chih Chang, and Jeff Erickson for useful discussions and feedback. We especially thank Thomas Church for helping us clarify some of the topological issues involved and suggesting how to modify the proof of Thm. 14 to extend it to the S_n case (Thm. 18).

1 Introduction

A unique game is a constraint satisfaction problem in which every constraint is between two variables, say x_i and x_j, and for each assignment to x_i, there is a unique assignment to x_j which satisfies the constraint, and vice versa; in particular, the domain of each variable must have the same size k. Khot [21] conjectured that, for any $\varepsilon, \delta > 0$, there is some k such that it is NP-hard to distinguish between instances of Unique Games in which at most a δ fraction of constraints can be satisfied from those in which at least a $1 - \varepsilon$ fraction of the constraints can be satisfied. The Unique Games Conjecture (UGC) rose to prominence in the past 15 years partly because it implies that our current best approximation algorithms for many problems are optimal assuming $\mathsf{P} \neq \mathsf{NP}$ (e. g., [21, 25, 11, 23, 26]), thus explaining the lack of further progress on these problems. It is also interesting because, unlike $\mathsf{P} \neq \mathsf{NP}$, the UGC is a more well-balanced conjecture, with little consensus in the community as to its truth or falsehood. This even-handedness, together with the progress made in the last 10 years (e. g., [7, 34, 35, 27, 30, 3, 6, 24]) suggests that the UGC might be closer to resolution than other major conjectures in complexity theory like P versus NP or VP versus VNP.

Khot, Kindler, Mossel, and O'Donnell [23] showed that the UGC is equivalent to its special case, Γ-Max-2Lin, in which every constraint is of the form $x_i - x_j = c$, treated as equations over $\mathbb{Z}_k$ (the cyclic group of order k). This beautiful simplification might lead one to naively expect that the UGC is somehow primarily about linear algebra, but this is potentially misleading. Indeed, a key feature in the solution of linear systems of equations is the ability to perform Gaussian elimination by taking linear combinations of equations, but when the equations are not satisfiable, taking linear combinations of equations can significantly change the maximum fraction of equations that are satisfiable. This leads us to ask: is there a domain of classical mathematics – other than modern computer science – in which the UGC is naturally situated?

In this paper, we argue that (algebraic) topology is such a domain. The starting point for our investigation is Linial's observation [29][3] that the only two known "UGC-complete" problems – UG itself and Γ-Max-2Lin– are in fact instances of finding a maximum section of a (G-)covering space over the underlying constraint graph of the CSP (topological terminology will be explained in §2; $G = S_k$ for UG and $G = \mathbb{Z}_k$ for Γ-Max-2Lin). In the case of Γ-Max-2Lin, we observe that this is naturally equivalent to the well-studied 1-Homology Localization problem from computational topology (see, e. g., [14, 9, 12, 16, 13, 10, 39, 15]). We also observe that the reduction from Γ-Max-2Lin to UG [23] gives a well-defined map of G-covering spaces.

[3] In an earlier version of this paper we were unaware of Linial's observation, which appears on slides 55–56 of [29]. Once we were made aware of this, for which we thank an anonymous reviewer and Hsien-Chih Chang, we checked Linial's slides, and the first author remembered having *attended* the talk that Linial gave at MIT on 11 May 2005! This was, in fact, one of the first theory seminars the first author had ever attended, and at the time he certainly didn't know what bundles were, nor the UGC; he also could not recall whether Linial actually made it to those slides that particular day. We believe that Linial was the first to make this observation.

To cement the topological nature of the UGC, we then show that Maximum Section of a G-Covering Space, or 1-Homology Localization, on (cell decompositions of) 2-manifolds, rather than graphs, is still UGC-complete. This gives the first new UGC-complete problem in over a decade. Of course, there is some subjectivity as to what counts as a UGC-complete problem being "distinct" from UG itself. In particular, Γ-Max-2Lin can be viewed as UG with certain additional hypotheses satisfied, but nonetheless "feels" different (this difference can be made a little more precise topologically, see Remark 3). In our proof, we'll see that 1-Homology Localization on 2-manifolds can also be viewed as a special case of Γ-Max-2Lin satisfying certain additional hypotheses, but again, Homology Localization "feels different" to us. Regardless, our results draw what we believe is a new connection between UGC and computational topology.

The UGC-completeness of this problem also partially settles a question of Chen and Freedman [14] on the complexity of the 1-Homology Localization problem on 2-manifolds. In particular, Chen and Freedman [14, p. 438, just before § 4.3] asked whether it was hard to approximate 1-Homology Localization with coefficients in $\mathbb{Z}_2$ on 2-manifolds; while some of the details are left unspecified, given the context in their paper we may conservatively infer (see our discussion in § 5) that they were asking for inapproximability to within all constant factors for triangulations of 2-manifolds. We show:

- Assuming UGC, for any constant $\alpha > 1$, there is a k such that 1-Homology Localization over $\mathbb{Z}_k$ on cell decompositions of 2-manifolds cannot be efficiently approximated to within α. In particular, this problem is UGC-complete.
- Assuming UGC, for any $\varepsilon > 0$, there is a k such that 1-Homology Localization over $\mathbb{Z}_k$ on *triangulations* of 2-manifolds cannot be efficiently approximated to within $7/6 - \varepsilon$.

Although the above are our most technically demanding results, which are applications of UGC to 1-Homology Localization – and, in the course of this, showing a new UGC-complete problem – we hope that the connections we have drawn between UGC and computational topology will lead to further progress on both of these topics in the future.

Related work. Linial [29] first observed that UG could be phrased in terms of Maximum Section of a Graph Lift (though we were unaware of this when we began our investigations, see Footnote 3). To our knowledge, since Linial's observation there have been no other works relating approximation problems in computational topology with the Unique Games Conjecture, nor is there previous work on the problem of Maximum Section of G-Covering Spaces. In this paper we extend Linial's observation by showing that the reduction of [23] gives a well-defined map of covering spaces, and we relate UGC to the well-studied problem of Homology Localization.

Here we briefly survey related work on approximation problems in computational topology, particularly those related to Homology Localization and the question of Chen and Freedman that we partially answer. Note that d-Homology Localization fixes the dimension of the homology considered, but allows the input to consist of d-homology classes on manifolds of arbitrarily large dimension. In our paper we consider 1-Homology Localization on graphs and 2-manifolds. For a more comprehensive overview of the area, as well as more direct motivations for the problem of Homology Localization, see [14, Sections 1 and 3].

1-Homology Localization with coefficients in $\mathbb{Z}_2$ is NP-hard to optimize exactly on simplicial complexes [12] and even on 2-manifolds [9]. Chen and Freedman showed it was NP-hard to approximate 1-Homology Localization on triangulations of 3-manifolds to within all constant factors, and that it was NP-hard to approximate d-Homology Localization on triangulations of manifolds for any $k \geq 2$. The best known algorithms for 1-Homology

Localization on a 2-manifold are given in several papers by Chambers, Erickson, and Nayyeri [9, 16, 10] (see also [39]). In particular, [9] solve the problem in polynomial time for fixed genus; however, for triangulations of 2-manifolds, the genus $g = \Theta(e/v)$ (follows from Euler's formula), and for instances of Unique Games to be difficult one must have the edge density e/v growing without bound, so their result seems not to solve the instances of 1-Homology Localization relevant for the UGC. Dey, Hirani, and Krishnamoorthy [15] showed that Homology Localization *in the 1-norm* over $\mathbb{Z}$ can be done in polynomial time; in our paper we are primarily concerned with the 0-norm.

Organization. In § 2 we give preliminaries. § 3 contains the details of how to view Unique Games and Γ-Max-2Lin as instances of Maximum Section of a G-Covering Space, and the result that the KKMO reduction [23] gives a well-defined map of G-covering spaces. In § 4 we show that 1-Homology Localization on cell decompositions of 2-manifolds is UGC-complete. In § 5 we show how our techniques partially settle a question of Chen and Freedman, and in § 6 we discuss open questions. All omitted proofs are available in the full version, which also has a discussion of generalizations to non-Abelian groups G and arbitrary topological spaces X. Content that appears in both has the same numbers in both, and references to the full version are displayed as, e. g., Obs. 12^{FULL} or App. A^{FULL}.

2 Preliminaries

2.1 The Unique Games Conjecture and inapproximability

We refer to the textbooks [37, 5] for standard material on approximation algorithms and inapproximability, and to the survey [22] for more on the Unique Games Conjecture. Here we briefly spell out the needed definitions and one standard lemma that will be of use.

An instance of a constraint satisfaction problem (CSP) is specified by a set of variables $x_1, \ldots, x_n$, for each variable x_i a domain D_i (which we will always take to be a finite set, and, in fact, we will have all D_i equal to one another), and a set of constraints. Each constraint is specified by a subset $\{x_{i_1}, \ldots, x_{i_k}\}$ of k of the variables (each constraint may, in principle, have a different arity k), and a k-ary relation $R \subseteq D_{i_1} \times \cdots \times D_{i_k}$. An assignment to the variables satisfies a given constraint if the assignment is an element of the associated R.

A CSP may be specified by restricting the arity and type of relations allowed in its instances, as well as the allowed domains for the variables. The value function associated to a CSP is $v(x, s) =$ the fraction of constraints in x satisfied by s, and we get the associated maximization problem. Given a CSP $\mathcal{P}$, the associated *gap problem* $\mathsf{Gap}\mathcal{P}_{c,s}$ is the promise problem of deciding, given an instance x, whether $OPT(x) \leq s$ or $OPT(x) \geq c$. (An algorithm solving $\mathsf{Gap}\mathcal{P}_{c,s}$ may make either output if x violates the promise, that is, if $s < OPT(x) < c$.) In general, the parameters c, s may depend on the problem size $|x|$.

If the optimization problem $\mathcal{P}$ can be approximated to within a factor α by some algorithm, then essentially the same algorithm solves $\mathsf{Gap}\mathcal{P}_{c,s}$ whenever $c/s > \alpha$. In the contrapositive, if $\mathsf{Gap}\mathcal{P}_{c,s}$ is, for example, NP-hard, then so is approximating $\mathcal{P}$ to within a factor c/s. The converse is false.

▶ **Problem** (Unique Games)**.** The Unique Games problem with k colors, denoted $UG(k)$, is the CSP whose domains all have size exactly k, and where each constraint has arity 2 and is a bijection between the domains of its two variables.

The natural n-vertex graph associated to a UG instance – in which there is an edge (i, j) for each constraint on the pair (x_i, x_j) – is called its *constraint graph*. A $UG(k)$ instance is completely specified by its constraint graph, together with a a permutation $\pi_{ij} \in S_k$ on each edge (i, j), specifying the constraint that, for $i < j$, $x_i = \pi_{ij}(x_j)$.

▶ **Conjecture** (Khot [21], Unique Games Conjecture (UGC)). *For all $\varepsilon, \delta > 0$, there exists a $k \in \mathbb{N}$ such that* Gap$UG(k)_{1-\varepsilon,\delta}$ *is* NP*-hard.*

Since the community is divided on this exact conjecture, the UGC is sometimes interpreted more liberally as saying that Gap$UG(k)_{1-\varepsilon,\delta}$ is *somehow* hard, for example, not in P, BPP, or quasiP. All our results will work equally well under any of these interpretations, so we often just refer to "efficient approximation" or write "approximating ... is hard."

A polynomial-time *gap-preserving reduction* from Gap$\mathcal{P}_{c,s}$ to Gap$\mathcal{Q}_{c',s'}$ (say, both minimization or both maximization problems) is a polynomial-time function f such that $OPT_{\mathcal{P}}(x) \leq s \Rightarrow OPT_{\mathcal{Q}}(f(x)) \leq s'$ and $OPT_{\mathcal{P}}(x) \geq c \Rightarrow OPT_{\mathcal{Q}}(f(x)) \geq c'$. If $\mathcal{P}$ is a maximization problem and $\mathcal{Q}$ is a minimization problem, then a gap-preserving reduction is similarly an f such that $OPT_{\mathcal{P}}(x) \leq s \Rightarrow OPT_{\mathcal{Q}}(f(x)) \geq c'$ and $OPT_{\mathcal{P}}(x) \geq c \Rightarrow OPT_{\mathcal{Q}}(f(x)) \leq s'$.

We say informally that a problem $\mathcal{P}$ is "UGC-complete" if there are gap-preserving reductions from Gap$\mathcal{P}_{\alpha,\beta}$ to GapUG$_{1-\varepsilon,\delta}$ and GapUG$_{1-\varepsilon,\delta}$ to Gap$\mathcal{P}_{\alpha,\beta}$ (where, in one direction, ε, δ may depend on α, β, and vice versa in the other direction) such that some UGC-like statement holds for $\mathcal{P}$ – such as "For any $\alpha < 1$, $\beta > 0$ Gap$\mathcal{P}_{\alpha,\beta}$ is hard to approximate" – if and only if UGC holds. Prior to this paper, the only known UGC-complete problems were UG itself,[4] and Γ-Max-2Lin(q) [23]:

▶ **Problem** (Max-2Lin(A) and Γ-Max-2Lin(A)). Let A be an Abelian group. Max-2Lin(A), or Max-2Lin(k) when $A = \mathbb{Z}_k$, consists of those instances of UG where every variable has A as its domain, and each constraint takes the form $ax_i + bx_j = c$ for some $a, b \in \mathbb{Z}$ and $c \in A$ (not necessarily the same a, b, c for all constraints). Γ-Max-2Lin(A) is the same, except that all the constraints have the form $x_i - x_j = c$ for some $c \in A$ (not necessarily the same for all constraints).

We will use the following standard lemma, which allows one to add a small number of new constraints to a given graph in a way that preserves an inapproximability gap.

▶ **Lemma 1.** *For a class $\mathcal{A}$ of graphs, let $UG_{\mathcal{A}}$ denote the Unique Games Problem on graphs from $\mathcal{A}$. Given two classes of graphs $\mathcal{A}, \mathcal{B}$, let $f : \mathcal{A} \to \mathcal{B}$ be a polynomial-time computable function such that for all $G \in \mathcal{A}$, $E(G) \subseteq E(f(G))$ and $|E(f(G)) \setminus E(G)| = O(v)$ where v is the number of vertices in G of degree ≥ 1. If the number of edges added is at most av, then there is a gap-preserving reduction from $UG_{\mathcal{A},1-\varepsilon,\delta}$ to $UG_{\mathcal{B},1-\varepsilon_0,\delta_0}$ where $\varepsilon_0 = \varepsilon + \Delta$ and $\delta_0 = \delta + \Delta$, for any $1 > \Delta > 2\delta a/(1 + 2\delta a)$ (in particular, with $\Delta \to 0$ as $\delta \to 0$).*

In particular, if $UG_{\mathcal{A}}$ is UGC-hard, then so is $UG_{\mathcal{B}}$. The same holds with "UG" everywhere replaced by Max-2Lin or Γ-Max-2Lin.

The intuition here is that one can always satisfy a number of constraints linear in the number of vertices (just choose a spanning tree or forest), so adding another linear number of constraints will only affect the inapproximability gap by a constant, which is negligible as δ and ε get arbitrarily small. For completeness the full version contains its (easy) proof §2.1$^{\text{FULL}}$.

2.2 G-covering spaces of graphs

▶ **Definition 2** (Graph lifts, a.k.a covering graph). Let X be a graph. A *graph lift*, or *covering graph*, is another graph Y with a map $p\colon V(Y) \to V(X)$ that such that the restriction of p

[4] And slight variants, for example UG on bipartite graphs [21], or a variant due to Khot and Regev [25] in which one tries to maximize the number of vertices in an induced subgraph all of whose constraints are satisfied, rather than simply maximizing the number of constraints satisfied.

to the neighborhood of each $v \in V(Y)$ is a bijection onto the neighborhood of $p(v) \in X$. If X is connected, then the number k of points in $p^{-1}(v)$ is independent of v, and we say Y is a *k-sheeted cover* of X. The set of vertices $p^{-1}(v)$ is called the *fiber* over v.

Graph lifts have found many uses in computer science and mathematics, particularly in the study of expanders (e. g., [8, 20, 31, 32, 2]) and, via the next example, Unique Games.

▶ **Example 3** (Label-extended graph)**.** Given an instance of Unique Games with constraint graph X on vertex set $[n]$ and domain size k, and permutations π_e on the directed edges $e \in E(X)$, its *label-extended graph* is a graph with vertex set $[n] \times [k]$, and with an edge from (v, i) to (w, j) iff $\pi_{v,w}(i) = j$. In particular, the label-extended graph is a k-sheeted covering graph of X.

In our setting, all of our covering graphs will come naturally with a group that acts on their fibers, and we would like to keep track of this group action, for reasons that will become clear in § 3. For example, the label-extended graph of a UG instance carries a natural action of S_k on each fiber (as would be the case with *any* k-sheeted covering graph), and the label-extended graph of a Max-2Lin(A) instance has a natural action of the Abelian group A on each fiber. From the point of view of approximation, keeping track of the group currently seems of little relevance, but it may be useful from the topological point of view, so we state our definitions and results carefully keeping track of the (monodromy) group.

▶ **Definition 4** (G-covering graph, see [18] and [1, Definition 12][5])**.** Let G be a group of permutations on a set of size k. A G-covering space of a graph X is a k-sheeted covering graph $Z = (V(X) \times [k], E)$ such that the permutations on each edge come from the action of the group G. Symbolically, for each edge $(u, v) \in X$, there is a group element $g_{u,v} \in G$ and Z contains an edge from (u, i) to (v, j) iff $g_{u,v}(i) = j$.

In topological terminology, this definition is equivalent to a "G-bundle with finite fibers" or to a covering space of the graph whose monodromy group (the group generated by considering the permutations you get by going around cycles in the graph) is contained in G.

We consider a graph X as a 1-dimensional geometric simplicial complex in the natural way, in which each edge has length 1.

▶ **Definition 5** (Section of a covering graph)**.** Given a covering graph $p\colon Y \to X$, a *section* of p is a continuous map $s\colon X \to Y$ (of topological spaces, as above) such that $p(s(x)) = x$ for all x. That is, it is a choice, for each $x \in X$, of a unique point in $p^{-1}(x)$, in a way that varies continuously with x.

▶ **Example 6.** Consider a covering graph $p\colon Y \to X$ where X is a triangle ($V(X) = \{0, 1, 2\}$, $E(X) = \{\{1, 2\}, \{0, 1\}, \{0, 2\}\}$), Y is a 6-cycle with vertex set $\{0, \ldots, 5\}$, in its natural ordering (edge set $\{\{i, i + 1 \pmod 6\} : i \in \{0, \ldots, 5\}\}$), and $p(i) = i \pmod 3$. This covering

[5] Note that here we consider G *as a permutation group* – that is, technically, an abstract group *together* with an action on a set of size k, as is done in Definition 12 of the preprint [1]. In [2, Definition 1] (and [1, Definition 1]) they define a "G-lift" for an abstract group G as a G-covering graph where the action of G is the regular action on itself by left translations. To translate between this terminology, that of bundles, and that of voltage graphs [18], we have:

Action	Covering graph	Bundle	Lift	Voltage Graph
regular	regular covering space	principal G-bundle	G-lift	ordinary voltage graph
general	G-covering graph (not nec. regular)	general G-bundle with finite fibers	$(G, S, \cdot)$-lift	permutation voltage graph

graph has no section: For, without loss of generality, we may suppose it has a section s and $s(0) = 0$. Then by continuity (imagine dragging a point $x \in Z$ from the point 0 across the edges of G), $s(1) = 1$ and $s(2) = 2$. But then as we continue varying our point in X across the edge $\{2, 0\}$, we find that we must also have $s(0) = 3$, a contradiction. If we think of Y as "lying over" X in the manner specified by p, we see that it is the label-extended graph of the UG instance on X with domain size 2, and π_e being the unique transposition for every edge e. The fact that there is no section here corresponds precisely to the fact that the UG instance is not completely satisfiable; see Obs. 11.

Given two covering graphs $p_i \colon Y_i \to X$ $(i = 1, 2)$ of the same graph X, a *homomorphism* between covering graphs is a continuous map $f \colon Y_1 \to Y_2$ such that $p_2 \circ f = p_1$. In particular, this means that the points in Y_1 in the fiber over $x \in X$ (that is, in $p_1^{-1}(x)$) are mapped to points in Y_2 that also lie in the fiber over the same point x.

▶ **Example 7** (Isomorphism of label-extended graphs)**.** Given two instances of UG on the same constraint graph X, if their label-extended graphs are isomorphic as covering graphs of X, then there is a natural bijection between assignments to the variables in the two instances which precisely preserves the number of satisfied constraints. Indeed, such an isomorphism is nothing more than re-labeling the elements of the domain of each variable.

▶ **Observation 8.** *Given two G-covering graphs $p_\ell \colon Y_\ell \to X$ $(\ell = 1, 2)$ with edge permutations $\pi_{ij}^{(\ell)}$, any isomorphism of covering graphs between them has the following form: for each $i \in V(X)$ there is a permutation π_i (not necessarily in G) such that $\pi_{ij}^{(2)} = \pi_i^{-1} \pi_{ij}^{(1)} \pi_j$.*

Conversely, given a G-covering space $p_1 \colon Y_1 \to X$ with edge permutations π_{ij}, and an element $g_i \in G$ for each $i \in V(X)$, the G-covering space Y_2 defined by $\pi_{ij}^{(2)} = g_i \pi_{ij}^{(1)} g_j^{-1}$ is isomorphic to Y_1.

▶ **Definition 9.** An *isomorphism of G-covering graphs* $p_\ell \colon Y_\ell \to X$ $(\ell = 1, 2)$ is an isomorphism of covering graphs such that the π_i (notation from Obs. 8) can be chosen to lie in G. When we say two G-covering spaces are "isomorphic", we mean isomorphic as G-covering spaces (not just as covering spaces).

All these notions generalize from graphs to topological spaces (see App. A$^{\text{FULL}}$).

2.3 Homology and cohomology in dimensions ≤ 2

Let us briefly recall the problem of 1-Homology Localization, specialized to our context. Given a simplicial complex, or more generally a combinatorial CW complex X (see §2.3$^{\text{FULL}}$) of dimension 2, the group of d-cycles ($d = 0, 1, 2$) with coefficients in an Abelian group A, denoted $C_d(X, A)$ is isomorphic to the group A^{n_d}, where n_d is the number of d-simplices in X ($d = 0$: vertices, $d = 1$: edges, $d = 2$: triangles or 2-cells). We identify the coordinates of such a vector with an assignment of an element of A to each d-simplex of X. The support of a d-chain $a \in C_d(X, A)$ is the set of d-simplices that appear in a with nonzero coefficient. The boundary of a 1-simplex $[i, j]$ is the 0-chain $\partial_1([i, j]) := [i] - [j]$, and this operator ∂_1 is extended to a function $C_1(X, A) \to C_0(X, A)$ by A-linearity. Similarly, the boundary of a 2-cell $[i_1, i_2, \ldots, i_\ell]$ is the 1-cycle $\partial_2[i_1, i_2, \ldots, i_\ell] := [i_1, i_2] + [i_2, i_3] + [i_3, i_4] + \cdots + [i_{\ell-1}, i_\ell] - [i_1, i_\ell]$, and we extend this to a map $C_2(X, A) \to C_1(X, A)$ by A-linearity. When no confusion may arise, we may refer to both of these maps simply as ∂. The image of the boundary map ∂_d is a subgroup of $C_{d-1}(X, A)$, denoted $B_{d-1}(X, A)$.

A *d-cycle* is a d-chain $a \in C_d(X, A)$ such that $\partial a = 0$. For example, if X is a graph, a 1-cycle is just a union of cycles, in the usual sense of cycles in a graph; if X is a 2-manifold,

the only 2-cycles are A-scalar multiples of the entire manifold; all vertices are 0-cycles. The d-cycles form a subgroup of $C_d(X, A)$ denoted $Z_d(X, A)$.

Two d-cycles that differ by the boundary of a $(d+1)$-cycle are *homologous*. The d-homology classes form the quotient group $H_d(X, A) := Z_d(X, A)/B_d(X, A)$. $H_0(X, A) \cong A^c$ where c is the number of connected components, and if X is a closed 2-manifold then $H_2(X, A) = A$. For closed 2-manifolds, thus the main interest is in $H_1(X, A)$.

▶ **Problem** (1-Homology Localization, 1-HomLoc)**.** Given a simplicial complex (resp., combinatorial CW complex) X and a 1-cycle $a \in Z_1(X, A)$, determine the sparsest homologous representative of a, that is, a 1-cycle a' homologous to a with minimum support among all 1-cycles homologous to a.

Cohomology is, in a sense, dual to homology. A d-cochain on X with coefficients in an Abelian group A is a homomorphism $C_d(X, \mathbb{Z}) \to A$; equivalently, it is determined by its values (from A) on the d-simplices (or d-cells) of X. The d-cochains form a group $C^d(X, A) \cong A^{n_d}$, where n_d is the number of d-simplices or d-cells. Given a d-cochain $f\colon X_d \to A$ (X_d being the d-simplicies or d-cells of X), its *coboundary* is the function $(\delta f)\colon X_{d+1} \to A$ defined by $(\delta f)(\Delta) = f(\partial\Delta)$ for any $(d+1)$-cell Δ, and then extended A-linearly. Thus $\delta f \in C^{d+1}(X, A)$. A *$d$-cocycle* is a d-cochain whose coboundary is zero, equivalently, a d-cochain that evaluates to 0 on the boundary of any $(d+1)$-chain. The d-cocycles form a subgroup $Z^d(X, A) \leq C^d(X, A)$. A d-coboundary is the coboundary of some $(d-1)$-cochain; these form a subgroup $B^d(X, A) \leq Z^d(X, A)$. Two d-cochains that differ by a d-coboundary are said to be *cohomologous*, and the cohomology classes form a group $H^d(X, A) := Z^d(X, A)/B^d(X, A)$. As with homology, for 2-manifolds the main cohomological interest is in H^1. The support of a d-cocycle is the number of d-simplices to which it assigns a nonzero value.

▶ **Problem** (1-Cohomology Localization, 1-CohoLoc(G))**.** Let G be a group (see App. A.3$^{\text{FULL}}$ for the definitions in the non-Abelian case). Given a simplicial complex (resp. combinatorial CW complex) X and a 1-cocycle $a \in Z^1(X, G)$, find the sparsest cohomologous representative.

On closed surfaces, we have the following equivalence between these problems:

▶ **Observation 10** (J. Erickson, personal communication)**.** *1-Cohomology Localization on CW complexes that are closed surfaces is equivalent to 1-Homology Localization on CW complexes that are closed surfaces, and dually (swapping the order of homology and cohomology).*

The proof essentially follows from (the proof of) Poincaré Duality; see §2.4$^{\text{FULL}}$.

3 The only known UGC-complete problems are Maximum Section of a G-Covering Graph

In this section we carefully write out the proof of Linial's observation [29] that UG (and Γ-Max-2Lin) is a special case of the following topological problem. We further observe that the reduction between these two problems [23] preserves the topological covering space structure of these problems.

▶ **Problem** (Maximum Section of a (G-)Covering Graph)**.** Let G be a group of permutations. Given a graph X and a G-covering graph $p\colon Y \to X$, find the partial section of p that is defined on as many edges of X as possible.

▶ **Observation 11** (Linial [29, pp. 55–56]). *$UG(k)$ is the same as the Maximum Section of a S_k-Covering Graph Problem. Furthermore, given two isomorphic S_k-covering spaces of the same constraint graph X, there is a bijection between assignments to the two instances of UG that exactly preserves the number of constraints satisfied.*

Proof. Given an instance of $UG(k)$ on constraint graph X, we have seen in Example 3 that the label-extended graph Y of this instance is a covering graph of X; let $p\colon Y \to X$ be the natural projection. As every constraint is a permutation in S_k, (p, Y) is clearly an S_k-covering space of X. Now suppose $u\colon X \to [k]$ is an assignment to the variables of X. We claim that this assignment can be extended to a section of p over the subset of $E(X)$ consisting precisely of those edges of X corresponding to constraints satisfied by u. For suppose $(i, j) \in E(X)$ and the constraint on (i, j) is satisfied by u, that is, $\pi_{ij}(u(i)) = u(j)$. Then in Y, there is an edge from $(i, u(i))$ to $(j, u(j))$ by construction. We may thus extend u to send points of the edge $(i, j) \in E(X)$ to the points of the edge $((i, u(i)), (j, u(j)) \in E(Y)$ bijectively and continuously, and thus extend u to a section of p that is defined over any edge satisfied by u.

Conversely, suppose $s\colon X' \to Y$ is a section of the restriction of p to $X' \subseteq X$, that is, $p|_{X'}\colon p^{-1}(X') \to X'$. We may use s to define a partial assignment to the variables of X. Namely, for any $i \in V(X')$ (that is, $i \in V(X)$ and $i \in X'$), define $u(i)$ by the equation $s(i) = (i, u(i)) \in V(Y)$. We claim that any edge of X contained entirely in X' is satisfied by this assignment u. Indeed, suppose the edge $(i, j) \in E(X)$ is contained entirely in X'. As s assigns $u(i)$ to i and $u(j)$ to j, and is continuous over all of X', there must be an edge from $s(i) = (i, u(i))$ to $s(j) = (j, u(j))$ in Y. But this is the same as saying that $\pi_{ij}(u(i)) = u(j)$, and thus the constraint on this edge is satisfied. Therefore, maximizing the cardinality of the number of edges over which a section of p exists is the same as maximizing the cardinality of the number of constraints satisfied.

Finally, it is a folklore result that there is a bijection between assignments to two instances of $UG(k)$ on the same constraint graph X whose label-extended graphs are isomorphic graph lifts. Indeed, such an isomorphism corresponds simply to re-labeling the domain of each variable. As S_k is the maximal permutation group on a set of size k, Obs. 8 says that two isomorphic G-covering spaces of X are isomorphic graph lifts. ◀

▶ **Observation 12** (cf. Linial [29]). *Let A be an Abelian group. The Γ-Max-2Lin(A) Problem is the same as the Maximum Section of an A-Covering Graph Problem, where we view A as a permutation group acting on itself by translations. Furthermore, given two isomorphic A-covering spaces of the same constraint graph X, there is a bijection between assignments to the two instances of UG that exactly preserves the number of constraints satisfied.*

The proof is essentially the same as above; see Obs. 12^{FULL} for a little more detail.

Khot, Kindler, Mossel, and O'Donnell [23] prove that Γ-Max-2Lin(q) is UG-hard by giving a gap-preserving reduction from UG. Our next result is that this reduction sends isomorphic S_k-covering spaces to isomorphic $\mathbb{Z}_q$-covering spaces. See Prop. 13^{FULL} for the proof.

▶ **Proposition 13.** *The reduction [23] from $UG(k)$ to Γ-Max-2Lin(q) gives a well-defined map {isomorphism classes of S_k-covering spaces} $\to$ {isomorphism classes of $\mathbb{Z}_q$-covering spaces}.*

▶ Remark. It is well-understood that one difference between $UG(k)$ and Γ-Max-2Lin(k) is that if an instance of Γ-Max-2Lin(k) is satisfiable, then one can choose an arbitrary vertex, assign an arbitrary value in $\mathbb{Z}_k$ to this vertex, and propagate this value across the entire graph

using the constraints, and this will always be a solution. In contrast, even if an instance of $UG(k)$ is satisfiable, starting from a given vertex there may be (in the worst case) only one value that can be assigned to that vertex in such a way that the propagated assignment is actually satisfying.

In topological language, the above translates nearly exactly as follows (see Footnote 5): A principal bundle is trivial (a direct product) if and only if it has a section and if so, such a section can be found by making an arbitrary choice at one vertex and propagating. We note that when A is Abelian and the action is faithful, every A-covering space is a principal A-bundle, whereas this need not be the case for non-Abelian groups.

4 1-Homology Localization on cell decompositions of 2-manifolds is UGC-complete

▶ **Theorem 14.** *1-Homology Localization on cell decompositions of closed orientable surfaces is UGC-complete. More precisely, the Unique Games Conjecture holds if and only if for any $\varepsilon, \delta > 0$, there is some $k = k(\varepsilon, \delta)$ such that* Gap*1-HomLoc*$_{1-\varepsilon,\delta}$ *on cell decompositions of closed orientable surfaces with coefficients in $\mathbb{Z}_k$ is* NP*-hard.*

To make the equivalence here a bit cleaner, we introduce a small variant of Γ-Max-2Lin on surfaces instead of graphs:

▶ **Problem** (Γ-Max-2Lin(A) on surfaces)**.** Let A be an Abelian group. Given a cell decomposition of a closed surface X, and a 1-cocycle on X with coefficients in A treat the 1-cocycle as defining an instance of Γ-Max-2Lin(A). In other words, this problem is the same as Γ-Max-2Lin(A) on the 1-skeleton X_1 of X, except that we only consider instances of Γ-Max-2Lin(A) in which the sum of the constraints along each cycle of X_1 that is the boundary of a 2-cell of X is zero.

From the second characterization in the definition ("in other words..."), it is clear that Γ-Max-2Lin(A) on cell decompositions of surfaces is potentially easier than Γ-Max-2Lin(A) on graph. We show that Γ-Max-2Lin(A) on surfaces nonetheless remains UGC-complete.

Proof of Thm. 14. The proof proceeds as follows, and will take up the remainder of this section: Γ-Max-2Lin(A) on graphs $\leq$ Γ-Max-2Lin(A) on surfaces $\cong$ 1-CohoLoc on surfaces $\cong$ 1-HomLoc on surfaces, where all reductions here are gap-preserving reductions. The first reduction is the technically tricky part, embodied in Prop. 17 below. The second equivalence is Obs. 15, which we do first since it will inform some of our subsequent discussion. The final equivalence is Obs. 10. ◀

▶ **Observation 15.** Γ*-Max-2Lin*(A) *on a cell decomposition of a surface X is equivalent (under gap-preserving reductions) to 1-CohoLoc*(A) *on the same cell decomposition of the same surface.*

Proof. Given an instance of Γ-Max-2Lin(A) specified by constants $a_{ij} \in A$ (that is, with constraints $x_i - x_j = a_{ij}$ for all edges in the cell decomposition of X), by the definition of Γ-Max-2Lin(A) on surfaces the function $a\colon (i,j) \mapsto a_{ij}$ is a 1-cocycle. We claim that the following is a natural bijection between assignments to the variables and cohomologous 1-cocycles such that the set of constraints satisfied by an assignment is precisely the complement of the support of the associated 1-cocycle: Given an assignment α_i to the variables x_i, treat α as a 0-cochain, and consider the 1-cocycle $a - \delta\alpha$. Note that α satisfies some edge (i,j) iff $(a - \delta\alpha)(i,j) = 0$, for $(a - \delta\alpha)(i,j) = a_{ij} - \alpha_i + \alpha_j$.

Conversely, given a cohomologous 1-cocycle a', the difference $a - a'$ is the coboundary of some 0-cochain: $a - a' = \delta\alpha$; treat α as an assignment to the variables. Using the same equation as before, we see that the support of a' is precisely the complement of the set of constraints satisfied by α.

Thus maximizing the number of satisfied constraints is equivalent to minimizing the support of a cohomologous cocycle. All that remains to check is that the equivalence above is indeed gap-preserving, noting that Γ-Max-2Lin(A) is a maximization problem while 1-CohoLoc(A) is a minimization problem. Here we take the value of a cocycle to be the *fraction* of nonzero edges. The above shows that if the maximum fraction of satisfiable constraints in the Γ-Max-2Lin(A) instance is ρ, then the minimum fraction of edges in the support of a cohomologous cocycle is $1 - \rho$ (since the number of edges in the same in both instances). So if a $\geq 1 - \varepsilon$ fraction of the constraints are satisfiable, then there is a cohomologous cocycle with support consisting of $\leq \varepsilon$ fraction of the edges; and if a $\leq \delta$ fraction of the constraints are satisfiable, then every cohomologous cocycle contains a $\geq 1 - \delta$ fraction of the edges. ◀

Our strategy for the reduction from Γ-Max-2Lin(A) on graphs to Γ-Max-2Lin(A) on surfaces will be to take an arbitrary graph and embed it as the 1-skeleton of a closed surface in polynomial time, in a gap-preserving way. The complication is that we must be careful about adding 2-cells. Given an instance of Γ-Max-2Lin(A) on a graph X, and some cycle in X such that the sum of the constraints around the cycle is nonzero, we cannot simply "fill in" the cycle with a 2-cell. If we added such a cell to the complex, then the instance would no longer correspond to a 1-cocycle. Thus, we may only add 2-cells to cycles that are satisfiable in the given instance.

Furst, Gross, and McGeoch [17] give a polynomial-time algorithm to find the *maximal genus embedding* of a graph, that is, an embedding on an orientable closed surface such that the complement of the graph decomposes into a disjoint union of disks, in such a way as to maximize the genus of the surface. In a cellular embedding, Euler's polyhedral formula applies: $V - E + F = 2 - 2g$, where V, E, and F are the numbers of vertices, edges, and faces of the complex, and g is the genus of the surface. Holding V and E fixed, we see that the problem of maximizing the genus is equivalent to minimizing the number of faces – and hence minimizing the additional properties an instance of Γ-Max-2Lin(A) on a surface must satisfy compared to being on a graph. In the extreme case, their algorithm may find an embedding with only one face, which "wraps around" the surface touching every edge twice, once on each side. Because the orientations of the region on either side oppose each other, the boundary of this region will always be zero. Therefore, if such a region is added to the complex, *any 1-cochain will still be a 1-cocycle.*

The algorithm of Furst, Gross, and McGeoch [17] is based on the work of Xuong [38], who gave a complete characterization of how many regions one needs to embed a graph. We restate the special case of his main result in which only one region is needed.

▶ **Theorem 16** (Xuong [38]). *A connected graph G has a one-face cellular embedding into a closed orientable surface if and only if there exists a spanning tree T such that every connected component of $G \setminus T$ has an even number of edges.*

See [17, Lemmas 3.1 & 3.3] for a proof of this special case.

We are now ready to formally describe our reduction and prove its correctness.

▶ **Proposition 17.** *There is a reduction of the form in Lemma 1 from Γ-Max-2Lin(A) on graphs to Γ-Max-2Lin(A) on cell decompositions of surfaces, and therefore Γ-Max-2Lin(A) on surfaces is UGC-complete.*

Proof. Suppose we are given a 1-cocycle on a graph X (every 1-cochain on a graph is a 1-cocycle, since there are no 2-cells). We describe an algorithm to transform X into a graph X' that admits a cellular embedding with one region. First, as X is the constraint graph of a Γ-Max-2Lin(A) instance, we may assume without loss of generality that X has no isolated vertices. Now, if X is disconnected, connect the components together, using one edge for each extra component. If the total number of edges is now odd, append a leaf attached by an edge to any vertex. Finally, add a "universal" vertex u, with an edge to every other vertex. Each new edge added in the preceding steps can be labeled with any constraint; it will not matter, but say $x_i - x_j = 0$ for concreteness. Let X' be the resulting graph. Letting T be the star spanning tree centered at u – that is, consisting of precisely the edges incident on u – we see that $X' \setminus T$ has only one component, with an even number of edges, so X' can be embedded in polynomial time with one region. The fact that this reduction preserves the inaproximability gap is the content of Lemma 1: If Γ-Max-2Lin(A) was hard on arbitrary graphs, it will still be hard on graphs with one-face embeddings, as this reduction added at most $(\frac{v}{2} - 1) + 1 + (v + 1) = O(v)$ edges. ◀

This completes the proof of Thm. 14. We would like to further reduce the instance so that it is a simplicial complex, rather than just a cell decomposition, which would prove that the simplicial version of the problem is UGC-hard as well. However, to break a 1-region embedding into triangles seems to require so many edges that it seems impossible to preserve the inapproximability gap completely. In the next section, we manage to preserve a 7/6 gap.

With G-covering spaces suitably defined, we show essentially the same result for non-Abelian G. We use G-covering spaces rather than 1-CohoLoc because 1-CohoLoc only corresponds to principal G-covering spaces, in which G acts on itself by translations. Using the following theorem, we could have shown UGC-completeness of Maximum Section of a G-Covering Space on cell decompositions of 2-manifolds without going through Max-2Lin; we chose the above route as the concepts with Abelian coefficient groups are simpler and more well-known. For the needed non-Abelian definitions and for the proof of this next result, see App. A$^{\text{FULL}}$. The hypothesis of the following result is satisfied by the symmetric groups S_k and all finite simple groups [33, 28].

▶ **Theorem 18.** *Let G be a group such that every product of commutators of G is equal to a commutator. Then Maximum Section of G-Covering Spaces on graphs reduces to Maximum Section of G-Covering Spaces on cell decompositions of 2-manifolds. In particular, the latter problem for (S_k-)covering spaces of cell decompositions of 2-manifolds is UGC-complete.*

5 On a question of Chen and Freedman

> "This raises the open question [of] whether localizing a one-dimensional class of a 2-manifold is NP-hard to approximate..." –Chen and Freedman [14, p. 438]

Note that Chen and Freedman showed that d-Homology Localization, for any fixed $d \geq 2$ is indeed NP-hard to approximate to within any constant factor for triangulations of manifolds (of unbounded dimension), as well as 1-HomLoc for triangulations of 3-manifolds. One might thus infer from the above quote that they were asking the same question – inapproximability to within any constant factor – for triangulations of 2-manifolds. We note that although a greedy algorithm gives a k-approximation to Unique Games over $\mathbb{Z}_k$ [36, Appendix], when we translate this maximization problem to the minimization problem of 1-HomLoc, we have essentially no control over the approximation ratio. We nonetheless show some inapproximability, by a different method (see §5$^{\text{FULL}}$ for the proof).

▶ **Theorem 19.** *Assuming UGC, for any $\varepsilon > 0$, there is some $k = k(\varepsilon)$ such that it is hard to approximate 1-HomLoc over $\mathbb{Z}_k$ on triangulations of surfaces to within a factor of $7/6 - \varepsilon$.*

Elsewhere in their paper they consider cell decompositions. If we relax their question to cell decompositions rather than triangulations, and we allow the coefficient group to be $\mathbb{Z}_k$ (rather than just $\mathbb{Z}_2$), then Thm. 14 states that their question becomes equivalent to UGC.

6 Future directions

Although our most technically demanding results were applying UGC to 1-Homology Localization – and, in the course of this, showing a new UGC-complete problem – we hope that the connections we have drawn between UGC and computational topology will lead to further progress on both topics in the future. Here we highlight a few specific questions suggested by our investigations.

Inapproximability of 1-Homology Localization for triangulations of 2-manifolds? Although our results partially settle a question of Chen and Freedman [14, p. 438], we leave open the following questions:

▶ **Open Question 20.** Show (unconditionally) that there is a $c > 1$ such that it is NP-hard to c-approximate 1-Homology Localization over $\mathbb{Z}_2$ on triangulations of 2-manifolds.

▶ **Open Question 21.** Does UGC imply that for all $c > 1$, it is hard to c-approximate 1-Homology Localization on triangulations of 2-manifolds (over $\mathbb{Z}_k$ for $k = k(c)$)?

We note that our reduction from Thm. 19 does not provide a strong enough gap for these problems to be immediately answered by the known NP-hardness results for Max-2Lin [19, 4]. In particular, Håstad shows that $\mathsf{Gap}\text{Max-2Lin}(2)_{\frac{12}{16},\frac{11}{16}}$ is NP-hard (up to an additive arbitrary δ in the soundness and completeness), but if we plug these values into ε_0, δ_0 from our proof, we get a ratio of $\frac{92}{129} < 1$. A quick calculation shows that, to get any NP-hardness result (without UGC) from the proof of Thm. 19, we would need an inapproximability ratio for Max-2Lin of strictly greater than $\frac{12}{5}$, which is not provided by [19] nor [4]. Given the best upper bounds on approximating Max-2Lin(p) [4], which are just slightly less than p, such a ratio is not possible for $p = 2, 3$, but is possible already for $p = 5$.

***G*-covering spaces for other families of groups and group actions.** The viewpoint of G-covering spaces suggests it might be fruitful to consider instances of UG that correspond to S_n-covering spaces for other actions of S_n. For example, one might consider the action of S_n on unordered k-tuples $\binom{[n]}{k}$, or even on n-vertex graphs $2^{\binom{[n]}{2}}$ (here, each variable in the UG would have the set of n-vertex graphs as its domain). We note that although much of the structure of any such instance is governed by the permutation constraints, the approximability properties may change significantly by varying the action. The general linear groups $\mathrm{GL}_n(\mathbb{F}_q)$, acting on the vector space $\mathbb{F}_q^n$, as well as Schur functors thereof, or other representations of $\mathrm{GL}_n(\mathbb{F}_q)$, strike us as leading to other possibly interesting approximation problems deserving further study. If one is looking for hard instances of Unique Games, one must construct such covering spaces so that they are not (sufficiently good) expanders [7]; see [2] for results on the expansion of G-covering spaces.

References

1 Naman Agarwal, Karthekeyan Chandrasekaran, Alexandra Kolla, and Vivek Madan. On the expansion of group-based lifts. arXiv:1311.3268 [cs.DM], 2016.

2 Naman Agarwal, Karthekeyan Chandrasekaran, Alexandra Kolla, and Vivek Madan. On the expansion of group-based lifts. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2017, August 16-18, 2017, Berkeley, CA, USA*, pages 24:1–24:13, 2017. doi:10.4230/LIPIcs.APPROX-RANDOM.2017.24.

3 Naman Agarwal, Guy Kindler, Alexandra Kolla, and Luca Trevisan. Unique games on the hypercube. *Chic. J. Theoret. Comput. Sci.*, pages Article 1, 20, 2015. doi:10.4086/cjtcs.2015.001.

4 Gunnar Andersson, Lars Engebretsen, and Johan Håstad. A new way of using semidefinite programming with applications to linear equations mod p. *J. Algorithms*, 39(2):162–204, 2001. Originally appeared in SODA 1999. doi:10.1006/jagm.2000.1154.

5 Sanjeev Arora and Boaz Barak. *Computational complexity*. Cambridge University Press, Cambridge, 2009. A modern approach. doi:10.1017/CBO9780511804090.

6 Sanjeev Arora, Boaz Barak, and David Steurer. Subexponential algorithms for unique games and related problems. *J. ACM*, 62(5):Art. 42, 25, 2015. Originally appeared in FOCS 2010. doi:10.1145/2775105.

7 Sanjeev Arora, Subhash A. Khot, Alexandra Kolla, David Steurer, Madhur Tulsiani, and Nisheeth K. Vishnoi. Unique games on expanding constraint graphs are easy: Extended abstract. In *STOC '08: 40th Annual ACM Symposium on Theory of Computing*, pages 21–28, 2008. doi:10.1145/1374376.1374380.

8 Yonatan Bilu and Nathan Linial. Lifts, discrepancy and nearly optimal spectral gap. *Combinatorica*, 26(5):495–519, 2006. doi:10.1007/s00493-006-0029-7.

9 Erin W. Chambers, Jeff Erickson, and Amir Nayyeri. Minimum cuts and shortest homologous cycles. In *SoCG '09: Proceedings of the 25th Annual Symposium on Computational Geometry*, pages 377–385, 2009. doi:10.1145/1542362.1542426.

10 Erin W. Chambers, Jeff Erickson, and Amir Nayyeri. Homology flows, cohomology cuts. *SIAM J. Comput.*, 41(6):1605–1634, 2012. Originally appeared in STOC 2009. doi:10.1137/090766863.

11 Shuchi Chawla, Robert Krauthgamer, Ravi Kumar, Yuval Rabani, and D. Sivakumar. On the hardness of approximating multicut and sparsest-cut. *computational complexity*, 15(2):94–114, Jun 2006. Originally appeared in CCC 2005. doi:10.1007/s00037-006-0210-9.

12 Chao Chen and Daniel Freedman. Quantifying homology classes II: Localization and stability. arXiv:07092512 [cs.CG], 2007.

13 Chao Chen and Daniel Freedman. Measuring and computing natural generators for homology groups. *Comput. Geom.*, 43(2):169–181, 2010. doi:10.1016/j.comgeo.2009.06.004.

14 Chao Chen and Daniel Freedman. Hardness results for homology localization. *Discrete Comput. Geom.*, 45(3):425–448, 2011. Originally appeared in SODA 2010. doi:10.1007/s00454-010-9322-8.

15 Tamal K. Dey, Anil N. Hirani, and Bala Krishnamoorthy. Optimal homologous cycles, total unimodularity, and linear programming. *SIAM J. Comput.*, 40(4):1026–1044, 2011. Originally appeared in STOC 2010. doi:10.1137/100800245.

16 Jeff Erickson and Amir Nayyeri. Minimum cuts and shortest non-separating cycles via homology covers. In *SODA '11: Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1166–1176. SIAM, Philadelphia, PA, 2011.

17 Merrick L. Furst, Jonathan L. Gross, and Lyle A. McGeoch. Finding a maximum-genus graph imbedding. *J. ACM*, 35(3):523–534, 1988. doi:10.1145/44483.44485.

18 Jonathan L. Gross and Thomas W. Tucker. Generating all graph coverings by permutation voltage assignments. *Discrete Math.*, 18(3):273–283, 1977. doi:10.1016/0012-365X(77)90131-5.

19 Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001. Originally appeared in STOC 1997. doi:10.1145/502090.502098.

20 Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bull. Amer. Math. Soc. (N.S.)*, 43(4):439–561, 2006. doi:10.1090/S0273-0979-06-01126-8.

21 Subhash Khot. On the power of unique 2-prover 1-round games. In *STOC '02: 34th Annual ACM Symposium on Theory of Computing*, pages 767–775. ACM, 2002. doi:10.1145/509907.510017.

22 Subhash Khot. On the unique games conjecture (invited survey). In *CCC '10: 25th IEEE Conference on Computational Complexity*, pages 99–121, June 2010. doi:10.1109/CCC.2010.19.

23 Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for MAX-CUT and other 2-variable CSPs? *SIAM J. Comput.*, 37(1):319–357, 2007. Originally appeared in FOCS 2004. doi:10.1137/S0097539705447372.

24 Subhash Khot and Dana Moshkovitz. Candidate hard unique game. In *STOC '16: 48th Annual ACM Symposium on Theory of Computing*, pages 63–76, 2016. doi:10.1145/2897518.2897531.

25 Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within $2-\epsilon$. *J. Comput. System Sci.*, 74(3):335–349, 2008. Originally appeard in CCC 2003. doi:10.1016/j.jcss.2007.06.019.

26 Subhash A. Khot and Nisheeth K. Vishnoi. The unique games conjecture, integrality gap for cut problems and embeddability of negative-type metrics into ℓ_1. *J. ACM*, 62(1):8:1–8:39, 2015. Originally appeared in FOCS 2005. doi:10.1145/2629614.

27 Alexandra Kolla. Spectral algorithms for unique games. *Comput. Complexity*, 20(2):177–206, 2011. Originally appeared in CCC 2010. doi:10.1007/s00037-011-0011-7.

28 Martin W. Liebeck, E. A. O'Brien, Aner Shalev, and Pham Huu Tiep. The Ore conjecture. *J. Eur. Math. Soc. (JEMS)*, 12(4):939–1008, 2010. doi:10.4171/JEMS/220.

29 Nati Linial. Lifts of graphs. Slides of presentation, available at http://www.cs.huji.ac.il/~nati/PAPERS/lifts_talk.pdf, 2005.

30 Konstantin Makarychev and Yury Makarychev. How to play unique games on expanders. In Klaus Jansen and Roberto Solis-Oba, editors, *Approximation and Online Algorithms: 8th International Workshop, WAOA 2010, Liverpool, UK, September 9-10, 2010. Revised Papers*, pages 190–200, 2011. doi:10.1007/978-3-642-18318-8_17.

31 Adam W. Marcus, Daniel A. Spielman, and Nikhil Srivastava. Interlacing families I: Bipartite Ramanujan graphs of all degrees. *Ann. of Math. (2)*, 182(1):307–325, 2015. doi:10.4007/annals.2015.182.1.7.

32 Adam W. Marcus, Daniel A. Spielman, and Nikhil Srivastava. Interlacing families II: Mixed characteristic polynomials and the Kadison-Singer problem. *Ann. of Math. (2)*, 182(1):327–350, 2015. doi:10.4007/annals.2015.182.1.8.

33 Oystein Ore. Some remarks on commutators. *Proc. Amer. Math. Soc.*, 2:307–314, 1951. doi:10.2307/2032506.

34 Prasad Raghavendra. Optimal algorithms and inapproximability results for every CSP? In *STOC '08: 40th Annual ACM Symposium on Theory of Computing*, pages 245–254, 2008. doi:10.1145/1374376.1374414.

35 Prasad Raghavendra and David Steurer. Graph expansion and the unique games conjecture. In *STOC '10: 42nd Annual ACM Symposium on Theory of Computing*, pages 755–764, 2010. doi:10.1145/1806689.1806792.

36 Sartaj Sahni and Teofilo Gonzalez. P-complete approximation problems. *J. ACM*, 23(3):555–565, 1976. doi:10.1145/321958.321975.

37 Vijay V. Vazirani. *Approximation algorithms*. Springer-Verlag, Berlin, 2001.

38 Nguyen Huy Xuong. How to determine the maximum genus of a graph. *J. Combin. Theory Ser. B*, 26(2):217–225, 1979. doi:10.1016/0095-8956(79)90058-3.

39 Afra Zomorodian and Gunnar Carlsson. Localized homology. *Comput. Geom.*, 41(3):126–148, 2008. doi:10.1016/j.comgeo.2008.02.003.

Solving Large-Scale Minimum-Weight Triangulation Instances to Provable Optimality

Andreas Haas
Department of Computer Science, TU Braunschweig
Braunschweig, Germany
haas@ibr.cs.tu-bs.de

Abstract

We consider practical methods for the problem of finding a minimum-weight triangulation (MWT) of a planar point set, a classic problem of computational geometry with many applications. While Mulzer and Rote proved in 2006 that computing an MWT is NP-hard, Beirouti and Snoeyink showed in 1998 that computing provably optimal solutions for MWT instances of up to 80,000 *uniformly distributed* points is possible, making use of clever heuristics that are based on geometric insights. We show that these techniques can be refined and extended to instances of much bigger size and different type, based on an array of modifications and parallelizations in combination with more efficient geometric encodings and data structures. As a result, we are able to solve MWT instances with up to 30,000,000 uniformly distributed points in less than 4 minutes to provable optimality. Moreover, we can compute optimal solutions for a vast array of other benchmark instances that are *not* uniformly distributed, including normally distributed instances (up to 30,000,000 points), all point sets in the TSPLIB (up to 85,900 points), and VLSI instances with up to 744,710 points. This demonstrates that from a practical point of view, MWT instances can be handled quite well, despite their theoretical difficulty.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases computational geometry, minimum-weight triangulation

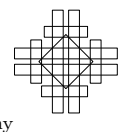
Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.44

Related Version A full version of this paper is available at `https://arxiv.org/abs/1802.06415`

Acknowledgements I want to thank Sándor Fekete and Victor Alvarez for useful discussions and suggestions that helped to improve the presentation of this paper.

1 Introduction

Triangulating a set of points in the plane is a classic problem in computational geometry: given a planar point set S, find a maximal set of non-crossing line segments connecting the points in S. Triangulations have many real-world applications, for example in terrain modeling, finite element mesh generation and visualization. In general, a point set has exponentially many possible triangulations and a natural question is to ask for a triangulation that is optimal with respect to some optimality criterion. Well known and commonly used is the Delaunay triangulation, which optimizes several criteria at the same time: it maximizes the minimum angle and minimizes both the maximum circumcircle and the maximum smallest enclosing circle of all triangles. Another natural optimality criterion, and the one we are considering in this paper is minimizing the total weight of the resulting triangulation, i.e., minimizing the sum of the edge lengths.



The minimum-weight triangulation (MWT) is listed as one of the open problems in the famous book from 1979 by Garey and Johnson on NP-completeness [10]. The complexity status remained open for 27 years until Mulzer and Rote [19] finally resolved the question and showed NP-hardness of the MWT problem.

Independently, Gilbert [11] and Klincsek [17] showed that, when restricting it to simple polygons, the MWT problem can be solved in $O(n^3)$-time with dynamic programming. The dynamic programming approach can be generalized to polygons with k inner points. Hoffmann and Okamoto [16] showed how to obtain the MWT of such a point set in $O(6^k n^5 \log n)$-time. Their algorithm is based on a polygon decomposition through x-monotone paths. Grantson et al. [12] improved the algorithm to $O(n^4 4^k k)$ and showed another decomposition strategy based on cutting out triangles [13] which runs in $O(n^3 k! k)$-time.

A promising approach are polynomial-time heuristics that either include or exclude edges with certain properties from any minimum weight triangulation. Das and Joseph [6] showed that every edge in a minimum weight triangulation has the *diamond property*. An edge e cannot be in MWT(S) if both of the two isosceles triangles with base e and base angle $\pi/8$ contain other points of S. Drysdale et al. [9] improved the angle to $\pi/4.6$. This property can exclude large portions of the edge set and works exceedingly well on uniformly distributed point sets, for which only an expected number of $O(n)$ edges remain. Dickerson et al. [8, 7] proposed the *LMT-skeleton heuristic*, which is based on a simple local-minimality criterion fulfilled by every edge in MWT(S). The LMT-skeleton algorithm often yields a connected graph, such that the remaining polygonal faces can be triangulated with dynamic programming to obtain the minimum weight triangulation.

Especially the combination of the diamond property and the LMT-skeleton made it possible to compute the MWT for large, well-behaved point sets. Beirouti and Snoeyink [3, 2] showed an efficient implementation of these two heuristics and they reported that their implementation could compute the exact MWT of 40,000 uniformly distributed points in less than 5 minutes and even up to 80,000 points with the improved diamond property.

Our contributions:

- We revisit the diamond test and LMT-skeleton based on Beirouti's and Snoeyink's [3, 2] ideas and describe several improvements. Our bucketing scheme for the diamond test does not rely on a uniform point distribution and filters more edges. For the LMT-skeleton heuristic we provide a number of algorithm engineering modifications. They contain a data partitioning scheme for a parallelized implementation and several other changes for efficiency. We also incorporated an improvement to the LMT-skeleton suggested by Aichholzer et al. [1].
- We implemented, streamlined and evaluated our implementation on various point sets. For the uniform case, we computed the MWT of 30,000,000 points in less than 4 minutes on commodity hardware; the limiting factor arose from the memory of a standard machine, not from the runtime. We achieved the same performance for normally distributed point sets. The third class of point sets were benchmark instances from the TSPLIB [20] (based on a wide range of real-world and clustered instances) and the VLSI library. These reached a size up to 744,710 points. This shows that from a practical point of view, wide range of huge MWT instances can be solved to provable optimality with the right combination of theoretical insight and algorithm engineering.

2 Preliminaries

Let S be a set of points in the euclidean plane. A triangulation T of S is a maximal planar straight-line graph with vertex set S. The weight $w(e)$ of an edge e is its euclidean length. A minimum-weight triangulation $\mathrm{MWT}(S)$ minimizes the total edge weight, i.e., $\sum_{e \in E(T)} w(e)$.

An edge e is *locally minimal* with respect to some triangulation $T(S)$ if either

(i) e is an edge on the convex hull of S, or

(ii) the two triangles bordering e in $T(S)$ form a quadrilateral q such that q is either not convex or e is the shorter diagonal of the two diagonals e and e' in q, i.e., $w(e) \leq w(e')$.

A triangulation T is said to be a *locally minimal triangulation* if every edge of T is locally minimal, i.e., the weight of T cannot be improved by edge flipping. A pair of triangles witnessing local minimality for some edge e in some triangulation is called a *certificate* for e. An *empty triangle* is a triangle that contains no other points of S except for its three vertices.

3 Previous tools

3.1 Diamond property

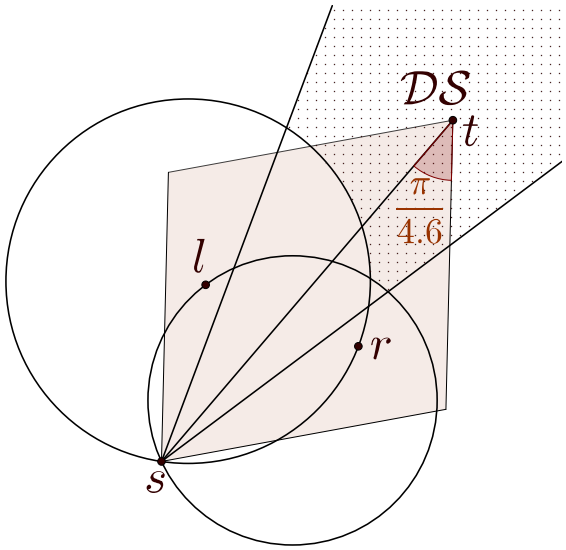


Figure 1 Points l and r induce a region $\mathcal{DS}$ such that all edges $e = st$ with $t \in \mathcal{DS}$ fail the diamond test. $\mathcal{DS}$ is called a dead sector (dotted area).

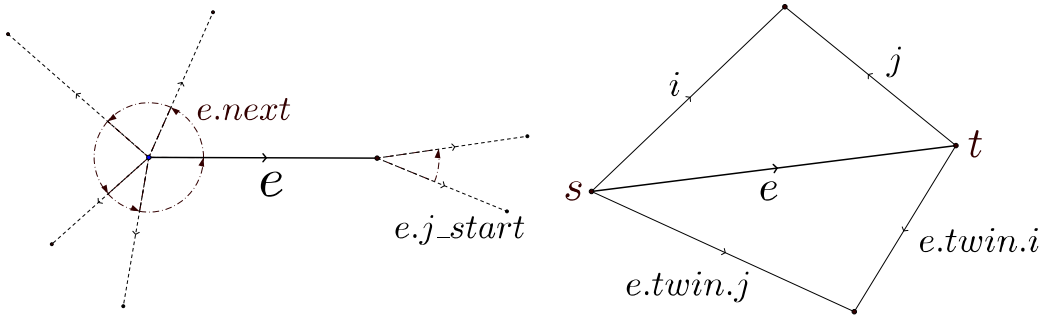
The diamond property can be used as a first step to exclude a large part of the edge set from any further consideration. A brute-force solution to test the diamond property for each edge takes $\Theta(n^3)$ time and is inefficient. To accelerate the test, Beirouti and Snoeyink [3] use a bucketing scheme based on a uniform grid with the grid size chosen such that on expectation a constant number of points lie in each cell. In order to quickly discard whole cells, they make use of *dead sectors*, which are illustrated in Figure 1. Suppose we want to test all edges with source s. If points l, r are already processed and known then all edges st with $t \in \mathcal{DS}$ will fail the diamond test, because l and r lie in the left, resp. right isosceles triangle. The boundary of a single dead sector depends on the angle and length of sl and sr; for multiple sectors it can be quite complicated. For each point s, all edges st that pass the

test are computed by first considering all other points in the cell containing s. Intersections of dead sectors with neighboring cells are computed and based on the result further cells are considered until all cells can be excluded by dead sectors.

3.2 LMT-skeleton

The LMT-skeleton was proposed by Dickerson et al. [8, 7], it is a subset of the minimum weight triangulation. The key observation is that MWT(S) is a locally minimal triangulation, i.e., no edge in MWT(S) can be flipped to reduce the total weight.

The LMT-skeleton algorithm eliminates all edges that have no certificate, i.e., for each edge e all pairs of empty triangles bordering e are examined until a certificate is found or no pairs are left. Eliminating e can invalidate previously found certificates. The remaining edges that are not intersected by any other remaining edge form the LMT-skeleton; they must be in all locally minimal triangulations.



(a) Half-edge e.

(b) i and j point to the current certificate edges.

Figure 2 Representation of half-edge e.

In order to avoid the $O(n^3)$ space required to store all empty triangles Beirouti and Snoeyink [3] propose a data structure based on half-edges. Half-edges store several pointers: a pointer to the target vertex and the twin half-edge; a pointer to the next edge in counterclockwise order around the source; and three additional pointers to scan for empty triangles (i, j, j_start); see Figure 2 for an illustration. A status flag indicates whether an edge is *possible, impossible* or *certain*. Furthermore, three additional pointers (*rightPoly, leftPoly, polyWeight*) are stored and used for the subsequent polygon triangulation step.

At the heart of the LMT-skeleton heuristic lies the `Advance` function, see Algorithm 1. `Advance` basically rotates edge i and j in counterclockwise order such that they form an

Algorithm 1: Advance. Adapted from [2] (Changed notation and corrected an error).

```
Function Advance(e)
    repeat
        while e.i.target is not left of e.j do
            e.i ← e.i.next ;
        while e.i.target is left of e.j do
            e.j ← e.j.next ;
    until e.i.target = e.j.target;
```

empty triangle if they point to the same vertex, i.e., $i.target = j.target$. Pointers i and j are initialized to $e.next$ resp. $e.j\_start$. The algorithm to find certificates is built on top of `Advance`. All pairs of triangles can be traversed by repeatedly calling `Advance` on half-edge e and e's twin in fashion similar to a nested loop. The "loop" is stopped when a certificate is found and can be resumed when the certificate becomes invalid. [2, 3]

After initializing the half-edge data structure, their implementation pushes all edges on a stack (sorted with longest edge on top) and then processes edges in that order. If for an edge e no certificate is found, an intersection test determines if e lies on the convex hull or if e is impossible. If e is detected to be impossible, a local scan restacks all edges with e in their certificate. After the stack is empty, all edges that remain possible and that have no crossing edges are marked as *certain*.

4 Our improvements and optimizations

4.1 Diamond property

For a uniformly distributed point set S with $|S| = n$ points, the expected number of edges to pass the diamond test is only $O(n)$. More precisely, Beirouti and Snoeyink [3] state that the number is less than $3\pi n/\sin(\alpha)$, where α is the base angle for the diamond property. We were able to tighten this value.

▶ **Theorem 1.** *Let S be a uniformly distributed point set in the plane with $|S| = n$ and let $\alpha \leq \pi/3$ be the base angle for the diamond property. Then the expected number of edges that pass the diamond test is less than $3\pi n/\tan(\alpha)$.*

Proof. Fix an arbitrary point $s \in S$ and consider the remaining points t_i , $0 \leq i \leq n-2$ in order of increasing distance to s. Edge $e_i := st_i$ fulfills the diamond property if at least one of the two corresponding isosceles triangles is empty, i.e., it contains non of the i points $t_0, \ldots, t_{i-1}$.

For any given distance r, each triangle has area $A = 1/4\tan(\alpha)r^2$. The points are uniformly distributed in the circle centered at s with radius r, thus the probability p that a fixed point lies in a fixed triangle is $p = A/\pi r^2 = \tan(\alpha)/4\pi$. Each triangle is empty with probability $(1-p)^i$. The whole diamond is empty with probability $(1-2p)^i$. It follows that at least one of the triangles is empty with probability $2(1-p)^i - (1-2p)^i$.

Let X_i be the indicator variable of the event that edge e_i fulfills the diamond property. Then $X = \sum_0^{n-2} X_i$ is the number of outgoing edges that pass the diamond test. By linearity of expectation and the geometric series, the expected value of X is bounded by

$$E[X] = \sum_{i=0}^{n-2} E[X_i] < \sum_{i=0}^{\infty} 2(1-p)^i - (1-2p)^i = \frac{2}{p} - \frac{1}{2p} = \frac{3}{2p} = \frac{6\pi}{\tan(\alpha)}$$

If we apply the same argument to each point in S, we are counting each edge twice. Hence the number of edges that pass the diamond test with base angle a is less than $3\pi n/\tan(\alpha)$. ◀

For $\alpha = \pi/4.6$ we get a value less than 11.5847, which is very close to the values observed and achieved by our implementation; see Table 1 in Section 5. In contrast, the value achieved by the implementation of Beirouti and Snoeyink is ≈ 14.3 [3].

4.2 Dead sectors and bucketing

Our bucketing scheme is based on the same idea of dead sectors as described by Snoeyink and Beirouti [3]. Our implementation differs in two points. Despite being simpler, it has

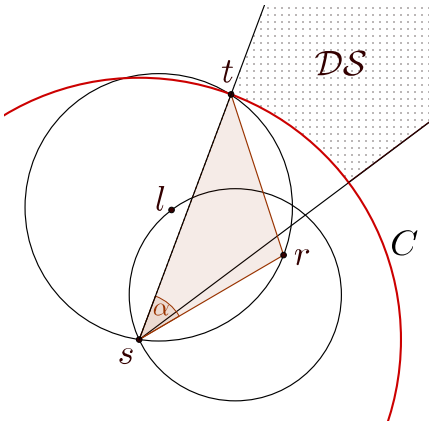


Figure 3 Simplified dead sector $\mathcal{DS}$ is bounded by two rays and circle C. C is induced by the longer of the two edges sl resp. sr and angle α.

higher accuracy and it can easily be integrated into common spatial data structures; such as quadtrees, kd-trees and R-trees. Therefore, it is not limited to uniformly distributed point sets.

In order to avoid storing complicated sector boundaries, we simplify the shape. Instead of bounding a sector $\mathcal{DS}$ by two circles as illustrated in Figure 1, we only use a single big circle C with center s at the expense of losing a small part of $\mathcal{DS}$. This allows a compact representation of dead sectors as a triple of three numbers: an interval consisting of two polar angles and a squared radius; see Figure 3.

The main ingredient for our bucketing scheme is a spatial search tree with support for incremental nearest neighbor searches, such as quadtrees, kd-trees or R-trees. A spatial search tree hierarchically subdivides the point set into progressively finer bounding boxes/rectangles until a predefined threshold is met. Incremental nearest neighbor search queries allow to traverse all nearest neighbors of a point in order of increasing distance. Such queries can easily be implemented by utilizing a priority queue that stores all tree nodes encountered during tree traversal together with the distances to their resp. bounding box (see Hjaltason and Samet [15]).

Pruning tree nodes whose bounding box lie in dead sectors is rather simple as follows: consider a nearest neighbor query for point s: when we are about to push a new node n into the priority queue, we compute the smallest polar angle interval I that encloses the bounding box of n and discard n if I is contained in the dead sectors computed so far. The interval of a bounding box is induced by the two extreme corners as seen from s, i.e., the leftmost and the rightmost corner.

Because nearest neighbors and tree nodes are processed in order of increasing distance, we can store sectors in two stages. On creation, they are inserted into a FIFO-queue; later only the interval component is inserted in a search filter used by the tree. The queue can be seen as a set of pending dead sectors with an attached activation distance δ. As soon as we process a point t with $\mathrm{d}(s,t) > \delta$ we can insert the corresponding interval into our filter.

This reduces the data structure used for the filter to a simple set of sorted non-overlapping intervals consisting of polar angles. Overlapping intervals are merged on insertion, which reduces the maximal number of intervals that need to be tracked at the same time to a very small constant[1].

[1] The exact value is 15 in our case, but it depends on an additional parameter and implementation details.

This leaves the issue of deciding which points are used to construct dead sectors. We store all points encountered during an incremental search query in an ordered set N sorted by their polar angle with respect to s. Every time we find a new point t, it is inserted into N and dead sectors are computed with the predecessor and the successor of t in N. There is no need to construct sectors with more than the direct predecessor and successor, because sectors between all adjacent pairs of points in N were already constructed on earlier insertions. Computing the activation distance for new sectors only requires a single multiplication of the current squared distance to t with a precomputed constant. Additionally, the diamond property of edge st is tested against a subset of N.

If we apply the above procedure to every single point, we generate each edge twice, once on each of the two endpoints. Therefore, we output only those edges $e = st$ such that $s < t$, i.e., s is lexicographically smaller than t. As a consequence, we can exclude a part of the left half-space right from the beginning by inserting an initial dead sector $\mathcal{DS}_0 = (1/2\pi + \alpha, 3/2\pi - \alpha)$ at distance 0. Points in the two wedges $(1/2\pi, 1/2\pi + \alpha]$ and $[3/2\pi - \alpha, 3/2\pi]$ are specially treated because they are still useful to generate dead sectors for the right half-space.

In order to increase cache efficiency we store the point set in a spatially sorted array. The points are ordered along a Hilbert curve, but the choice of a particular space-filling curve is rather arbitrary. Our spatial tree implementation is a quadtree that is built on top of that array during the sorting step. Profiling suggests the memory layout of the tree nodes is not important. We apply the diamond test to every single point and we can freely choose the order in which we process them. The points are spatially sorted and processed in this order, which leads to similar consecutive search paths in the tree and therefore most nodes are already in the CPU cache.

In order to avoid the expensive transcendental `atan2` function for polar angle computations, we can use any function that is monotonic in the polar angle for comparisons between different angles. One such function, termed *pseudo-angle*, was described by Moret and Shapiro [18]. The basic idea is to measure arc lengths on the L_1 unit circle, instead of the L_2 unit circle. With some additional transformations, the function can be rewritten to $\text{sign}(y)(1 - x/(|x| + |y|))$, where we define $\text{sign}(0) =: 1$. This function has the same general structure as `atan2`: a monotonic increase in the intervals $[0, \pi]$, $(\pi, 2\pi)$ and a discontinuity at π, with a jump from positive to negative values. Additionally, it gives rise to a one-line implementation (see the full version), which gets compiled to branch-free code.

4.3 LMT-skeleton

For "nicely" distributed point sets, a limiting factor of the heuristic is the space required to store the half-edge data structure in memory. In order to save some space we removed three variables from the original description (*rightPoly, leftPoly, polyWeight*). They serve no purpose until after the heuristic, when they are used for the polygon triangulation step (therefore, reducing cache-efficiency and wasting space). For edges marked *impossible* (typically the majority), they are never used at all; for the remaining edges they can be stored separately as soon as needed. We further reduce storage overhead by storing all edges in a single array sorted by source vertex (also known as a *compressed sparse row graph*). As a consequence, spatial ordering of the vertices carries over to the edge array. All outgoing edges of a single vertex are still radially sorted. In addition to the statuses *possible*, *certain*, *impossible*, we store whether an edge lies on the convex hull.

As mentioned in Section 3, certificates are found by utilizing `Advance` in fashion of a nested loop. It is crucial to define one half-edge of each pair as the primary one to distinguish

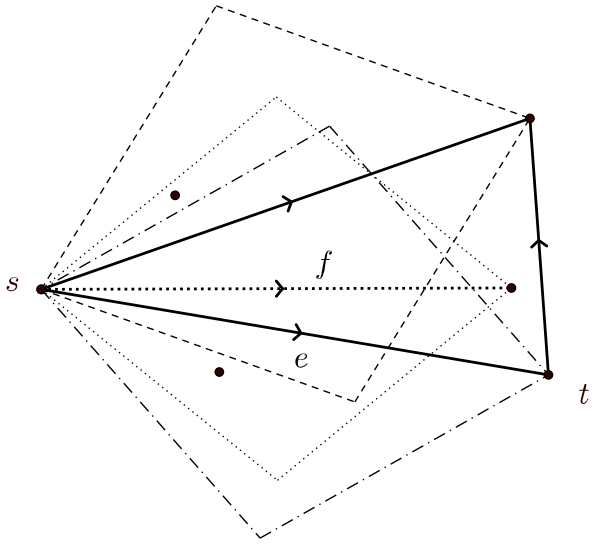


Figure 4 Edge f does not have the diamond property and in turn the `Advance` function fails: it stops with a non-empty triangle.

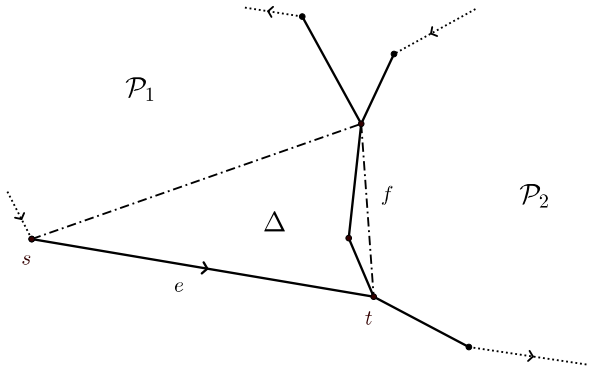


Figure 5 `Advance` can return non-empty triangle Δ which is part of two adjacent polygonal faces.

which half-edge corresponds to the outer resp. inner "loop". The choice is arbitrary as long as it is consistent throughout the execution of the algorithm.

Another problem that went unnoticed emerges when the diamond test and LMT-skeleton are combined. In this case `Advance` does not guarantee to find empty triangles; it may stop with non-empty triangles due to missing incident edges. An example is shown in Figure 4, where all edges with the exception of f pass the diamond test; calling `Advance` on e yields a non-empty triangle.

Fortunately, the side effect of wrong certificates is rather harmless. In the worst-case an otherwise impossible edge stays possible, which in turn may prevent other edges from being marked *certain*, however, no edge will incorrectly be marked *certain*. Even though wrong certificates occur frequently, we observed them to be of transient nature because some certificate edge itself becomes impossible later in the heuristic. Therefore, we still use `Advance` in our implementation to find certificates. However, it is important to keep in mind that the function can fail. Beirouti [2] states that they also use `Advance` to scan for empty triangles in simple polygons during the dynamic programming step after the LMT-skeleton. Even then `Advance` can fail by returning triangles that are part of two adjacent simple

Algorithm 2: Refactored LMT-skeleton algorithm.

```
begin
    Init data structures ;
    ST ← EmptyEdges(S)\ CH (S) ;                          /* Stack ST */
    LMT-Loop (ST) ;
    foreach Possible primary half-edge e do
        if ¬HasIntersections (e) then
            Mark e as certain;
Function LMT-Loop(ST)
    while ST is not empty do
        e ← Pop (ST) ;
        if e has no certificate then
            RestackEdges (e) ;    /* Push edges with e in their certificate.
             */
            Mark e as impossible;
```

polygonal faces; see Figure 5 for an example. In contrast to wrong certificates, this will lead to catastrophic results and cannot be ignored.

Pseudocode for our implementation is given in Algorithm 2. In essence it is still the same as given by Beirouti and Snoeyink [3], however, with some optimizations applied. First, the convex hull edges are implicitly given during initialization of the *j_start*-pointers and can be marked as such without any additional cost. Determining the convex hull edges beforehand allows to remove the case distinction inside the `LMT-Loop`, i.e., it removes all intersection tests that are applied to impossible edges. Secondly, sorting the stack by edge length destroys spatial ordering and the loss of locality of reference outweighs all gains on modern hardware. Without sorting, it is actually not necessary to push all edges onto the stack upfront. Lastly, with proper partitioning of the edges, the `LMT-Loop` can be executed in parallel – described in more detail in Section 4.4.

Additionally, we incorporated an improvement to the LMT-skeleton suggested by Aichholzer et al. [1]. Consider a certificate for an edge e, i.e., a quadrilateral q_e such that e is locally minimal w.r.t. q_e. It is only required that the four certificate edges $f_i \in q_e$ are not *impossible*, that is, edge f_i is either on the convex hull or in turn has some certificate q_i. Notice that q_i and q_e may not share a common triangle. However, if for edge f_i there is no such certificate q_i that shares a triangle with q_e, then edge e cannot be in any locally minimal triangulation and e can be marked *impossible*.

The improved LMT-skeleton is computationally much more expensive. Consider the case in which edge $e = (s, t)$ becomes impossible. In order to find invalid certificates, it is no longer sufficient to scan only those edges incident to either s or t. In addition to edges of the form (s, u), resp. (t, u), we also have to check all edges incident to any adjacent vertex u for invalid certificates. Because edges do not store the certificates for their certificate it gets even worse: we cannot know if an edge has to be restacked and we must restack and recheck all of them. Another consequence is that we cannot resume the traversal of triangles for any edge f_i, because we do not know where we stopped the last time.

We are left with a classic space-time trade-off and we chose not to store any additional data. Instead we apply the improved LMT-heuristic only to edges surviving an initial round of the normal LMT-heuristic.

4.4 Parallelization

Because the LMT-heuristic performs only local changes, most of the edges can be processed in parallel without synchronization. Problems occur only if adjacent edges are processed concurrently (for the improved LMT-skeleton this is unfortunately not true, because marking an edge *impossible* affects a larger neighborhood of edges). In order to parallelize the normal LMT-heuristic, we implemented a solution based on data partitioning without any explicit locking.

We cut the vertices V into two disjoint sets $V = V_1 \cup V_2$ and process only those edges with both endpoints in V_1 (resp. V_2) in parallel. Define X as the cut set $\{\{s,t\} \in E \mid s \in V_1, t \in V_2\}$, i.e., all edges with one endpoint in V_1 and the other in V_2. While edges in $E(V_1)$ resp. $E(V_2)$ are processed in parallel by two threads, edges in X are accessed read-only by both threads and are handled after both threads join. This way we never process two edges with a common endpoint in parallel.

This leaves the question of how to partition the vertices into two disjoint sets. Recall that all vertices are stored in contiguous memory and are sorted in Hilbert order. A split in the middle of the array partitions the points into two sets that are separated by a rather simple curve. Therefore, the cut set is likely to be small. Our half-edge array is sorted by source vertex, i.e., getting all edges with a specific source vertex in either half of the partition is trivial. Deciding if an edge $e = (s,t)$ is in the cut set consists of two comparisons of pointer t against the lower and upper bound of the vertex subset. Furthermore, with the fair assumption that the average degree of vertices is the same in both partitions, we obtain perfectly balanced partitions w.r.t. the number of edges.

In order to avoid a serial scan at the top, we push the actual work of computing X down to the leaves in the recursion tree. Scanning of the half-edge array starts at the leave nodes: processing of half-edges that belong to some cut set is postponed, instead they are passed back to the parent node. The parent in turn scans the edges it got from its two children, processes all edges it can and passes up the remaining ones. In other words, the final cut set X bubbles up in the tree, while all intermediate cuts are never explicitly computed. The edges passed up from a node typically contain half-edges of several higher-level cuts. This way, partitioning on each level of the recursion tree only takes constant time, while the actual work is fully parallelized at the leaf level.

Experiments and observations indicate that on large, uniformly distributed point sets approximately 0.15% of all edges make it back to the root node, i.e., the amount of serial processing is low and the approach scales well. On degenerate instances it can perform poorly; e.g. if all points lie on a circle, then half of the edges will be returned to the root. For such cases, the code could be extended to repartition the remaining edges with another set of cuts.

After the LMT-heuristic completes, we are left with many polygonal faces that still need to be triangulated. Our implementation traverses the graph formed by the edges with one producer thread in order to collect all faces and multiple consumer threads to triangulate them with dynamic programming.

5 Computational results

Computations were performed on a machine with an Intel i7-6700K quad-core and 64GB memory. The code was written in C++ and compiled with gcc 5.4.0.

We utilized CGAL [5] for its exact orientation predicates, however, parts of the code are still prone to numerical errors. For example, triangulating the remaining polygonal

Table 1 Diamond test implementation on uniformly distributed point sets. The table shows the mean and the standard deviation of 25 different instances. The extreme values are assumed by points at the point set boundary.

		Number of visited neighbors per point				
n	Edges	Mean	SD	Min	Max	$\mathcal{DS} = 2\pi$
10^1	36.16 ±2.63	9 ±0	0 ±0	9 ±0	9 ±0	0 ±0
10^2	882.8 ±27.69	55.6 ±3.1	16.6 ±2.04	23.72 ±4.82	98.56 ±1.27	30.4 ±4.61
10^3	10,731.7 ±159.9	72.52 ±1.56	23.16 ±1.3	22.68 ±4.55	173 ±14.61	737.84 ±10.91
10^4	$1.1316 \cdot 10^5$ ±471.24	77.64 ±0.69	26.64 ±0.73	19.08 ±2.3	363.72 ±20	9,126.08 ±18.74
10^5	$1.15 \cdot 10^6$ ±1,538.64	72.84 ±0.29	23.76 ±0.47	15.96 ±1.61	846.24 ±24.4	97,200.9 ±40.29
10^6	$1.1562 \cdot 10^7$ ±4,737.67	74 ±0.51	25.76 ±0.39	13.28 ±1.31	2,884.96 ±38.53	$9.9117 \cdot 10^5$ ±61.86
10^7	$1.1579 \cdot 10^8$ ±19,254	77 ±0.6	27.24 ±0.79	11.88 ±0.99	9,567.52 ±78.84	$9.9721 \cdot 10^6$ ±100.61
10^8	$1.1585 \cdot 10^9$ ±56,063.1	72 ±0.94	24.08 ±0.69	10.6 ±0.49	25,017.8 ±107.4	$9.9911 \cdot 10^7$ ±239.64

Table 2 LMT-skeleton statistics on uniformly distributed point sets.

	Possible edges after			Certain edges after		
n	Diamond	LMT	LMT+	LMT	LMT+	Simple Polygons
$1 \cdot 10^1$	36.76 ±2.78	3.8 ±3.84	3.72 ±3.62	19.32 ±2.22	19.32 ±2.22	0.68 ±0.61
$1 \cdot 10^2$	871.92 ±46.37	84.04 ±20.14	74.56 ±18.1	251.48 ±7.12	252.28 ±7.12	10.52 ±2.55
$1 \cdot 10^3$	10,687.4 ±146.68	1,150.32 ±98.05	1,031.96 ±86.46	2,540 ±32.33	2,548.04 ±31.41	128 ±9.2
$1 \cdot 10^4$	$1.1322 \cdot 10^5$ ±661.16	12,637 ±281.25	11,271.76 ±251.6	25,193.44 ±73.29	25,287.56 ±76.43	1,367.08 ±24.65
$1 \cdot 10^5$	$1.1503 \cdot 10^6$ ±1,696.31	$1.2941 \cdot 10^5$ ±1,198.41	$1.1523 \cdot 10^5$ ±973.14	$2.5129 \cdot 10^5$ ±322.29	$2.5227 \cdot 10^5$ ±306.72	13,819.44 ±67.93
$1 \cdot 10^6$	$1.1563 \cdot 10^7$ ±5,459.02	$1.3044 \cdot 10^6$ ±2,708.78	$1.1617 \cdot 10^6$ ±2,486.36	$2.5098 \cdot 10^6$ ±847.61	$2.5194 \cdot 10^6$ ±860.53	$1.3904 \cdot 10^5$ ±232.43
$1 \cdot 10^7$	$1.1579 \cdot 10^8$ ±17,587.01	$1.3074 \cdot 10^7$ ±11,021.75	$1.1645 \cdot 10^7$ ±8,825.57	$2.5088 \cdot 10^7$ ±2,774.11	$2.5184 \cdot 10^7$ ±2,727.23	$1.3931 \cdot 10^6$ ±607.95
$3 \cdot 10^7$	$3.4747 \cdot 10^8$ ±28,678.6	$3.9239 \cdot 10^7$ ±18,919.14	$3.4949 \cdot 10^7$ ±15,068.66	$7.5258 \cdot 10^7$ ±4,637.8	$7.5547 \cdot 10^7$ ±4,563.03	$4.1797 \cdot 10^6$ ±969.6

faces requires to compute and compare the sum of radicals, which we implemented with double-precision arithmetic. For small instances, it was possible to compare the results of our implementation against an independent implementation based on an integer programming formulation of the MWT problem. However, straightforward integer programming becomes infeasible quite fast and comparisons for point sets with thousands of points were not possible.

5.1 Uniformly and normally distributed point sets

Table 1 shows results of our diamond test implementation on uniformly distributed point sets with sizes ranging from 10 to 10^8 points. The table shows the mean values and the standard deviation of 25 different instances. Each instance was generated by choosing n points uniformly from a square centered at the origin. Point coordinates were double-precision values. The diamond test performs one incremental nearest neighbor query for each point in order to generate the edges that pass the test. On average only a small number of neighbors need to be processed for each point. The last column shows the number of queries where all nodes in the spatial tree were discarded because dead sectors covered the whole search space. The numbers show that this is the regular case; the exceptional cases occur at points near the point set "boundary".

Table 2 shows statistics for the LMT-heuristic on uniformly distributed point sets. The instance sizes range from 10 points up to 30,000,000 points. For each size 25 different instances were generated. For the largest instances, the array storing the half-edges consumes nearly 39 GB of memory on its own. The serial initialization of the half-edge data structure, which basically amounts to radially sorting edges, takes longer than the parallel `LMT-Loop` on uniformly and normally distributed points. The improved LMT-skeleton by Aichholzer et al. is denoted LMT+ in the tables. The resulting skeleton was almost always connected in the computations and the number of remaining simple polygons that needed to be triangulated is

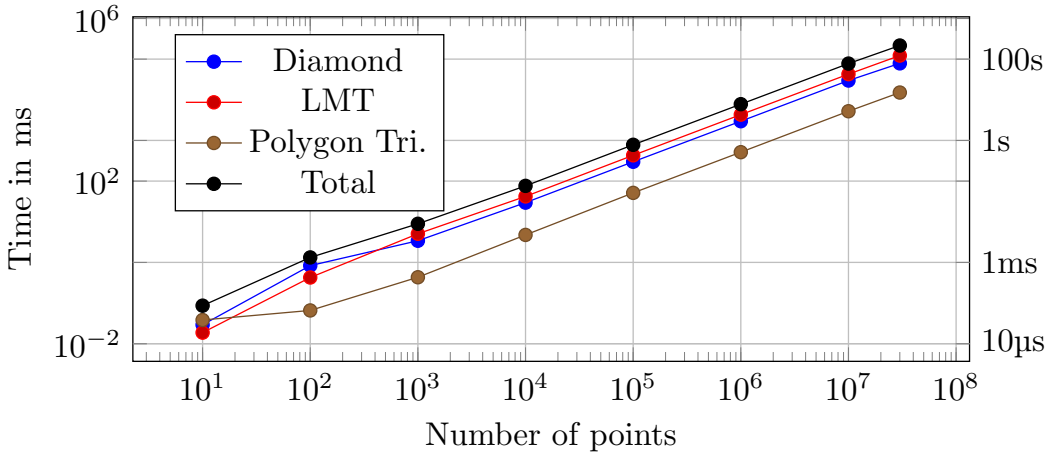


Figure 6 LMT-skeleton runtime on uniformly distributed point sets.

shown in the last column. Only one instance of size $3 \cdot 10^7$ contained one non-simple polygon in the experiments. While developing and optimizing the implementation, we computed the LMT-skeleton of several hundred instances with a million points, without ever seeing a disconnected component.

As we can see, the LMT-skeleton eliminates most of the possible edges with only $\approx 11\%$ remaining. Given that any triangulation has $3n - |\texttt{CH}| - 3$ edges, the certain edges amount to $\approx 83\%$ of the complete triangulation. The improved LMT-skeleton reduces the amount of possible edges by another 10%, but it provides hardly any additional certain edges.

The results on normally distributed point sets are basically identical. Point coordinates were generated by two normally distributed random variables $X, Y \sim \mathcal{N}(\mu, \sigma^2)$, with mean $\mu = 0$ and standard deviation $\sigma \in \{1, 100, 100000\}$. The tables are given in the full version.

5.2 TSPLIB + VLSI

In addition to uniformly and normally distributed instances, we ran our implementation on instances found in the well-known TSPLIB [20], which contains a wide variety of instances with different distributions. The instances are drawn from industrial applications and from geographic problems. All 94 instances have a connected LMT-skeleton and can be fully triangulated with dynamic programming to obtain the minimum weight triangulation. The total time it took to solve all instances of the TSPLIB was approximately 8.5 seconds. A complete breakdown for each instance is given in the full version.

Additional point sets can be downloaded at `http://www.math.uwaterloo.ca/tsp/vlsi/`. This collection of 102 TSP instances was provided by Andre Rohe, based on VLSI data sets studied at the Forschungsinstitut für Diskrete Mathematik, Universität Bonn. The LMT-heuristic is sufficient to solve all instances, except `lra498378`, which contained two non-simple polygonal faces. A complete breakdown is given in the full version. Our implementation of the improved LMT-skeleton performs exceedingly bad on some of these instances; see Table 3. These instances contain empty regions with many points on the "boundary". Such regions are the worst-case for the heuristics because most edges inside them have the diamond property, which in turn leads to vertices with very high degree. Whenever an edge is found to be impossible by the improved LMT-skeleton, almost all edges are restacked and rechecked. Given the overall poor results of the improved LMT-skeleton, storing additional data to increase performance and/or limiting it to non-simple polygons may be reasonable.

Table 3 Statistics for VLSI instances with long runtime.

	Time in ms					
Instance	Total	DT	LMT-Init	LMT-Loop	LMT+	Dyn. Prog.
ara238025	15,325	4,954	446	496	9,279	148
lra498378	382,932	44,267	1,238	7,532	329,292	599
lrb744710	484,430	7,952	1,377	2,661	471,564	872
sra104815	1,937	559	191	198	922	65

6 Conclusion

We have shown that despite of the theoretical hardness of the MWT problem, a wide range of large-scale instances can be solved to optimality.

Difficulties for other instances arise from two sources. On one hand, we have instances containing more or less regular k-gons with one or more points near the center. These configurations can lead to a highly disconnected LMT-skeleton (an example is given by Belleville et al. [4]) and require exponential time algorithms to complete the MWT. Preliminary experiments suggest that such configurations are best solved with integer programming. The example point set given by Belleville et al. [4] can easily be solved with CPLEX in less than a minute, while the dynamic programming implementation of Grantson et al. [14] was not able to solve it within several hours. On the other hand, we have instances containing empty regions with many points on their "boundary", such as empty k-gons and circles. They may be solvable in polynomial time, but trigger the worst-case behavior of the heuristics. Deciding what is the best approach to handle these two types of difficulties and integrating it into our implementation is left for future work.

References

1 Oswin Aichholzer, Franz Aurenhammer, and Reinhard Hainz. New Results on MWT Subgraphs. *Inf. Process. Lett.*, 69(5):215–219, 1999.

2 Ronald Beirouti. A Fast Heuristic for Finding the Minimum Weight Triangulation. Master's thesis, University of British Columbia, Vancouver, BC, Canada, Canada, 1997.

3 Ronald Beirouti and Jack Snoeyink. Implementations of the LMT Heuristic for Minimum Weight Triangulation. In Ravi Janardan, editor, *Symposium on Computational Geometry*, pages 96–105. ACM, 1998.

4 Patrice Belleville, J. Mark Keil, Michael McAllister, and Jack Snoeyink. On Computing Edges That Are In All Minimum-Weight Triangulations. In Sue Whitesides, editor, *Symposium on Computational Geometry*, pages 507–508. ACM, 1996.

5 CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

6 Gautam Das and Deborah Joseph. *Which triangulations approximate the complete graph?*, pages 168–192. Springer Berlin Heidelberg, Berlin, Heidelberg, 1989.

7 Matthew Dickerson, J. Mark Keil, and Mark H. Montague. A Large Subgraph of the Minimum Weight Triangulation. *Discrete & Computational Geometry*, 18(3):289–304, 1997.

8 Matthew Dickerson and Mark H. Montague. A (Usually?) Connected Subgraph of the Minimum Weight Triangulation. In Sue Whitesides, editor, *Symposium on Computational Geometry*, pages 204–213. ACM, 1996.

9 Robert L. Scot Drysdale, Scott A. McElfresh, and Jack Snoeyink. On exclusion regions for optimal triangulations. *Discrete Applied Mathematics*, 109(1-2):49–65, 2001.

10 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, 1979.

11 P. D. Gilbert. New results in planar triangulations. Master's thesis, University Illinois, 1979.

12 Magdalene Grantson, Christian Borgelt, and Christos Levcopoulos. A Fixed Parameter Algorithm for Minimum Weight Triangulation: Analysis and Experiments. Technical report, Lund University, Sweden, 2005.

13 Magdalene Grantson, Christian Borgelt, and Christos Levcopoulos. Minimum Weight Triangulation by Cutting Out Triangles. In Xiaotie Deng and Ding-Zhu Du, editors, *ISAAC*, volume 3827 of *Lecture Notes in Computer Science*, pages 984–994. Springer, 2005.

14 Magdalene Grantson, Christian Borgelt, and Christos Levcopoulos. Fixed Parameter Algorithms for the Minimum Weight Triangulation Problem. *Int. J. Comput. Geometry Appl.*, 18(3):185–220, 2008.

15 Gisli R. Hjaltason and Hanan Samet. Ranking in Spatial Databases. In *SSD '95: Proceedings of the 4th International Symposium on Advances in Spatial Databases*, pages 83–95, London, UK, 1995. Springer-Verlag.

16 Michael Hoffmann and Yoshio Okamoto. The Minimum Weight Triangulation Problem with Few Inner Points. In Rodney G. Downey, Michael R. Fellows, and Frank K. H. A. Dehne, editors, *IWPEC*, volume 3162 of *Lecture Notes in Computer Science*, pages 200–212. Springer, 2004.

17 G.T. Klincsek. Minimal Triangulations of Polygonal Domains. *Annals of Discrete Mathematics*, 9:121–123, 1980.

18 Bernard M. E. Moret and Henry D. Shapiro. *Algorithms from P to NP (Vol. 1): Design and Efficiency*. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 1991.

19 Wolfgang Mulzer and Günter Rote. Minimum-weight Triangulation is NP-hard. *J. ACM*, 55(2):11:1–11:29, 2008.

20 G. Reinelt. TSPLIB- a Traveling Salesman Problem Library. *ORSA Journal of Computing*, 3(4):376–384, 1991.

Dynamic Smooth Compressed Quadtrees

Ivor Hoog v.d.[1]
Dept. of Inform. and Computing Sciences, Utrecht University, the Netherlands
i.d.vanderhoog@uu.nl

Elena Khramtcova[2]
Computer Science Department, Université libre de Bruxelles (ULB), Belgium
elena.khramtsova@gmail.com

Maarten Löffler[3]
Dept. of Inform. and Computing Sciences, Utrecht University, the Netherlands
m.loffler@uu.nl

Abstract

We introduce dynamic smooth (a.k.a. balanced) compressed quadtrees with worst-case constant time updates in constant dimensions. We distinguish two versions of the problem. First, we show that quadtrees as a space-division data structure can be made smooth and dynamic subject to *split* and *merge* operations on the quadtree cells. Second, we show that quadtrees used to store a set of points in $\mathbb{R}^d$ can be made smooth and dynamic subject to *insertions* and *deletions* of points. The second version uses the first but must additionally deal with *compression* and *alignment* of quadtree components. In both cases our updates take $2^{\mathcal{O}(d \log d)}$ time, except for the point location part in the second version which has a lower bound of $\Omega(\log n)$; but if a pointer (*finger*) to the correct quadtree cell is given, the rest of the updates take worst-case constant time. Our result implies that several classic and recent results (ranging from ray tracing to planar point location) in computational geometry which use quadtrees can deal with arbitrary point sets on a real RAM pointer machine.

2012 ACM Subject Classification Theory of computation → Data structures design and analysis

Keywords and phrases smooth, dynamic, data structure, quadtree, compression, alignment, Real Ram

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.45

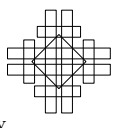
Related Version A full version is avaliable at [14], `https://arxiv.org/abs/1712.05591`.

Acknowledgements The authors would like to thank Joe Simons and Darren Strash for their inspiring initial discussion of the problem.

1 Introduction

The quadtree is a hierarchical spacial subdivision data structure based on the following scheme: starting with a single square, iteratively pick a square and subdivide it into four equal-size smaller squares, until a desired criterion is reached. Quadtrees and their higher-dimensional

[1] Supported by the Netherlands Organisation for Scientific Research (NWO) through project no 614.001.504.
[2] Supported by the SNF Early Postdoc Mobility grant P2TIP2-168563, Switzerland, and F.R.S.-FNRS, Belgium.
[3] Partially supported by the Netherlands Organisation for Scientific Research (NWO) through project no 614.001.504



equivalents have been long studied in computational geometry [26, 7, 17, 5, 11, 6, 18, 3] and are popular among practitioners because of their ease of implementation and good performance in many practical applications [12, 20, 4, 1, 9, 2]. We will not review the extremely rich literature here and instead refer to the excellent books by Samet [24] and Har-Peled [13].

Smooth and dynamic quadtrees

A quadtree is *smooth*[4] if each leaf is comparable in size to adjacent leaves. It has been long recognized that smooth quadtrees are useful in many applications [4], and smooth quadtrees can be computed in linear time (and have linear complexity) from their non-smooth counterparts [8, Theorem 14.4].

A quadtree is *dynamic* if it supports making changes to the structure in sublinear time. Recently, quadtrees have been applied in kinetic and/or uncertain settings that call for dynamic behaviour of the decomposition [9, 19, 15]. Bennett and Yap [3] show how to maintain a smooth *and* dynamic quadtree subject to amortized constant-time *split* and *merge* operations on the quadtree leaves.

At this point, it is useful to distinguish between the quadtree *an sich*, a combinatorial subdivision of space, and the quadtree as a data structure for storing a point set (or other set of geometric objects). Given a set P of points in the plane and a square that contains them, we can define the minimal quadtree that contains P to be the quadtree we obtain by recursively subdividing the root square until no leaf contains more than one (or a constant number of) point(s). It is well-known that such a minimal quadtree can have superlinear complexity, but can still be stored in linear space by using *compression* [8]. Additionally, when working in the Real RAM computation model, it may not be possible to keep different compressed components properly aligned [13, 18]. These complications imply that we cannot simply apply results known for standard/regular (henceforth called *uncompressed*) quadtrees. When maintaining a dynamic quadtree storing a point set P, we wish to support high-level operations of inserting points into P and removing points from P.[5]

Contribution

In this paper, we show that it is possible to maintain a quadtree storing a set of points P that is smooth and possibly compressed, which supports worst-case constant-time insertions and deletions of points into P, assuming we are given a pointer (*finger*) to the cell of the current quadtree containing the operation. Our result runs on a Real RAM pointer machine and generalises to arbitrary constant dimensions.

In the first half of the paper (Sections 3-5), we focus on the problem of making the quadtree itself dynamic and smooth, improving the recent result by Bennett and Yap [3] from amortized to worst-case constant time split and merge operations. The challenge here is to avoid cascading chains of updates required to maintain smoothness. Our key idea is to introduce several layers of smoothness: we maintain a core quadtree which is required to be 2-smooth, but cells added to satisfy this condition themselves need only be 4-smooth, etc (refer to Section 2 for the formal definition of 2^j-smoothness). In Section 3, we show that when defining layers in this way, we actually need only two layers in $\mathbb{R}^2$, and the second

[4] Also called *balanced* by some authors, which is not to be confused with the notion of balance in trees related to the relative weights of subtrees.

[5] In Table 1 in Section 2 we provide a complete list of operations and how they relate.

layer will always already be smooth. In Section 4, we show that we can handle updates on the core quadtree in constant time. In Section 5, we generalise the result to arbitrary dimensions (now, the number of layers, and thus the smoothness of the final tree, depends on the dimension).

In the second half of the paper (Sections 6-7), we focus on lifting our result to quadtrees that store a set of points on a pure real-valued pointer machine. The challenge here is to redefine compressed quadtrees in a consistent way across different layers of smoothness, and to re-align possibly misaligned components on the fly when such components threaten to merge. In Section 2, we show that we can view insertion of a point as a two-step procedure, where we first need to locate the correct leaf of the current tree containing the new point, and then actually insert it into the quadtree. We show in Section 6 that we can still handle the second step in worst-case constant time. In Section 7, we deal with the issue of avoiding the use of the floor operation, which is not available on a pure Real RAM.

Implications

Many publications in computational geometry use a concept which we shall dub **principal neighbor access**: The idea that for any cell C we can find its relevant neighbors in constant time:

▶ **Definition 1.** Given a quadtree T over $\mathbb{R}^d$, we say that we have **principal neighbor access** if for any cell C in T we can find the smallest cells C' in T with $|C'| \geq |C|$ and C' neighboring C in constant time if d is constant.

Bennet and Yap in [3] implement principal neighbor access by storing explicit **principle neighbor pointers** to the larger neighbors C'. Khramtcova and Löffler in [15] achieve principal neighbor access with the well known **level pointers** on a smooth quadtree. These two unique ways to guarantee principle neighbor access were also noted by Unnikrishnan *et al.* in [25] where a *threaded* quadtree in [25] maintained the equivalent to **principle neighbor pointers** as opposed to a *roped* quadtree in [23] which only maintained level pointers.

Principle neighborhood access allows us to traverse the neighborhood of any cell C in constant time. Bennet and Yap observe that any (non-compressed) quadtree must be smooth to dynamically maintain level pointers by using a sequence of cells that they insert. In the full version [14] we show that if quadtrees are compressed, you even need $\Theta(n)$ time for a single split operation to update the level pointers. For this reason Bennet and Yap develop their amortized-constant dynamic smooth quadtrees in $\mathbb{R}^d$ in [3]. We note that most applications that use principal neighbor access (dynamic variant of collision detection [21], ray tracing [1, 20] and planar point location [15, 19]) often run many operations parallelized on the GPU. In such an environment amortized analysis can become troublesome since there is a high probability that at least one GPU-thread obtains the worst-case $\mathcal{O}(n)$ running time. In that scenario the other threads have to wait for the slow thread to finish so the computations effectively run in $\mathcal{O}(n)$ time which makes our worst-case constant time algorithm a vast improvement.

A large number of papers in the literature explicitly or implicitly rely on the ability to efficiently navigate a quadtree, and our results readily imply improved bounds from amortized to worst-case [20, 15, 19, 9, 22], and extends results from bounded-spread point sets to arbitrary point sets [10]. Other papers could be extended to work for dynamic input with our dynamic quadtree implementation [1, 18, 21]. Several dynamic applications are in graphics-related fields and are trivially parallelizable, which enhanced the need for worst-case

bounds. In the full version [14] we give a comprehensive overview of the implications of our result.

2 Preliminaries

In this section we review several necessary definitions. Concepts that were already existing prior to this work are underlined. Consider the d-dimensional real space $\mathbb{R}^d$. For a hypercube $R \subset \mathbb{R}^d$, the **size** of R, denoted $|R|$, is the length of a 1-dimensional facet (i.e., an edge) of R.

▶ **Definition 2** (Quadtree)**.** Let R be an axis-aligned hypercube in $\mathbb{R}^d$. A **quadtree** T on the root cell R is a hierarchical decomposition of R into smaller axis-aligned hypercubes called **quadtree cells**. Each node v of T has an associated quadtree cell C_v, and v is either a leaf or it has 2^d equal-sized children whose cells subdivide C_v.[6]

From now on, unless explicitly stated otherwise, when talking about a quadtree cell C we will be meaning both the hypercube C and the quadtree node corresponding to C.

▶ **Definition 3** (Neighbor, Sibling neighbor)**.** Let C and C' be two cells of a quadtree T in $\mathbb{R}^d$. We call C and C' **neighbors**, if they are interior-disjoint and share (part of) a $(d-1)$-dimensional facet. We call C and C' **sibling neighbors** if they are neighbors and they have the same parent cell.

▶ Observation 1. Let C be a quadtree cell. Then: (i) C has at most $2d$ neighbors of size $|C|$; and (ii) For each of the d dimensions, C has exactly one sibling neighbor that neighbors C in that dimension.

▶ **Definition 4** (2^j-smooth cell, 2^j-smooth quadtree)**.** For an integer constant j, we call a cell C 2^j**-smooth** if the size of each leaf neighboring C is at most $2^j|C|$. If every cell in a quadtree is 2^j-smooth, the quadtree is called 2^j**-smooth**.

▶ Observation 2. If all the quadtree leaves are 2^j-smooth, then all the intermediate cells are 2^j-smooth as well.[7] That is, the quadtree is 2^j-smooth.

▶ **Definition 5** (Family related)**.** Let C_1, C_2 be two cells in a quadtree T such that $|C_1| \leq |C_2|$. If the parent of C_2 is an ancestor of C_1 we call C_1 and C_2 **family related**.[8]

We now consider quadtrees that store point sets. Given a set P of points in $\mathbb{R}^d$, and a hypercube R containing all points in P, an **uncompressed quadtree** that stores P is a quadtree T in $\mathbb{R}^d$ on the root cell R, that can be obtained by starting from R and successively subdividing every cell that contains at least two points in P into 2^d child cells.

▶ **Definition 6** (Compression)**.** Given a large constant α, an α-compressed quadtree is a quadtree with additional **compressed nodes**. A compressed node C_a has only one child C with $|C| \leq |C_a|/\alpha$, and the region $C_a \setminus C \subset \mathbb{R}^d$ does not contain any points in P. We call the link between C_a and C a **compressed link**, and C_a the **parent** of the compressed link.

Compressed nodes induce a partition of a compressed quadtree T into a collection of uncompressed quadtrees interconnected by compressed links. We call the members of such a collection the **uncompressed components** of T.

---

[6] We follow [18] in using *quadtree* in any dimension rather than dimension-specific terms (i.e. *octree*, etc).
[7] Observe that if a single (leaf) cell is 2^j-smooth, its ancestors do not necessarily have to be such.
[8] Observe that C_1 and C_2 do not have to be neighbors.

Table 1 Operations considered in this paper and the running times of the provided implementation.

Operation	Running time
I. Quadtree operations (uncompressed quadtree)	
Split a cell	$\mathcal{O}((2d)^d)$
Merge cells	$\mathcal{O}((2d)^d)$
II. Quadtree operations (α-compressed quadtree)	
Insert a component	$\mathcal{O}(d^2(6d)^d)$
Delete a component	$\mathcal{O}(d^2(6d)^d)$
Upgrowing of a component	$\mathcal{O}(\log(\alpha)d^2(6d)^d)$
Downgrowing of a component	$\mathcal{O}(\log(\alpha)d^2(6d)^d)$
III. Operations on the point set P, stored in a quadtree	
Insert a point into P	$\mathcal{O}(d\log(n) + \log(\alpha)d^2(6d)^d)$
Insert a point into P, given a finger	$\mathcal{O}(\log(\alpha)d^2(6d)^d)$
Delete a point from P	$\mathcal{O}(\log(\alpha)d^2(6d)^d)$

2.1 Quadtree operations and queries

Table 1 gives an overview of quadtree operations. It is insightful to distinguish three levels of operations. Operations on a compressed quadtree (II) internally perform operations on an uncompressed quadtree (I). Similarly, operations on a point set stored in a quadtree (III) perform operations on the compressed quadtree (II). We now give the formal definitions and more details.

▶ **Definition 7** (split, merge)**.** Given a leaf cell C of a quadtree T, the **split** operation for C inserts the 2^d equal-sized children of C into T. Given a set 2^d leaves of a quadtree T which share a parent, the **merge** operation removes these 2^d cells. The parent cell is now a leaf in T.

▶ **Definition 8** (upgrowing, downgrowing)**.** Let A be an uncompressed component of a compressed quadtree with a root R. **Upgrowing** of A adds the parent R' of R to T. The cell R' becomes the root of component A. **Downgrowing** of A removes the root R of A, and all the children of R except one child C. Cell C becomes the root of A. The downgrowing operation requires R to be an internal cell and all the points stored in A to be contained in one child C of R.

An insertion of a point p into the set P stored in an α-compressed quadtree T is performed in two phases: first, the leaf cell of T should be found that contains p; second, the quadtree should be updated. The first phase, called **point location**, can be performed in $\mathcal{O}(d\log(n))$ time (see edge oracle trees in [19]), and can be considered a query in our data structure. We refer to the second phase separately as **inserting a point given a finger**, see Table 1.

3 Static non-compressed smooth quadtrees in $\mathbb{R}^1$ and $\mathbb{R}^2$

We first view the quadtree as a standalone data structure subject only to merge and split operations. In this section we are given a unique non-smooth, uncompressed quadtree T_1 over $\mathbb{R}^1$ or $\mathbb{R}^2$ with n cells. It is known [8, Theorem 14.4] that an uncompressed quadtree can be made smooth by adding $\mathcal{O}(n)$ cells. However, the reader can imagine that if we want all the cells to be 2-smooth that we cannot make the quadtree dynamic with worst-case constant updates because balancing keeps cascading. In this section we show that if T_1 is

a quadtree over $\mathbb{R}^1$ or $\mathbb{R}^2$ then we can extend T_1 by consecutively adding $d \in \{1, 2\}$ sets of cells,[9] i. e., cells of d different *brands*, so that in the resulting *extended* quadtree T^* each cell is smooth according to its brand. The total number of added cells is $2^{\mathcal{O}(d \log(d))}n$.

3.1 Defining our smooth quadtree

We want to add a minimal number of cells to the original quadtree T_1 such that the cells of T_1 become 2-smooth and the balancing cells are smooth with a constant dependent on $d \in \{1, 2\}$. In general we want to create an **extended quadtree** T^* with $T_1 \subset T^*$ where all cells with brand j are 2^j-smooth for $j \leq d + 1$.

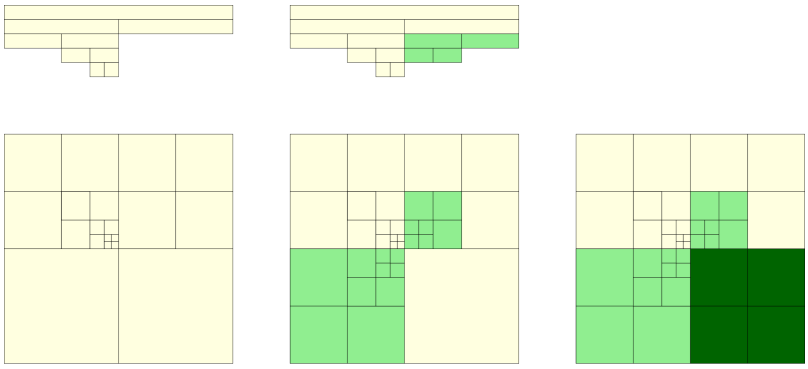


Figure 1 Left: a quadtree in $\mathbb{R}^1$ (up) and $\mathbb{R}^2$ (down); Center: the (light-green) cells of brand 2 added; Right: the (dark-green) cells of brand 3 added. In each row, the rightmost tree is the smooth version of the leftmost one.

The *true* cells (T_1) get brand 1. Figure 1 shows two quadtrees and its balancing cells. This example also illustrates our main result: to balance a tree T_1 over $\mathbb{R}^d$ we use $(d + 1)$ different brands of cells and the cells of the highest brand are automatically 2^{d+1}-smooth. This example gives rise to an intuitive, recursive definition for balancing cells in $\mathbb{R}^d$. In this definition we have a slight abuse of notation: For each brand j, we let T_j denote the set of cells with brand j, and T^j denote the quadtree associated with the cells in the sets T_i for all $i \leq j$:

▶ **Definition 9** (sets T_j)**.** Let T_1 be a set of true cells in $\mathbb{R}^d$. We define the sets $T_j, 2 \leq j \leq d+1$ recursively:

Given a set of cells T_j in $\mathbb{R}^d$, let T^j be the quadtree corresponding to $\cup_{i \leq j} T_i$.

We define the set T_{j+1} to be the minimal set of cells obtained by splitting cells of T^j, such that each cell in T_j is 2^j-smooth in T_{j+1}.

For each set T_j, to every cell in T_j we assign brand j.

▶ **Definition 10.** In $\mathbb{R}^d$, we define T^* to be T^{d+1}.

The extended quadtree T^* has three useful properties which we prove in the remainder of this section: the tree is unique, the size of the tree is linear in n, and cells in the tree which are related in ancestry have a related brand.

▶ **Lemma 11.** *For a given set T_j, the set T_{j+1} is unique.*

[9] In Section 5 we show the same is possible for quadtrees of arbitrary dimension d.

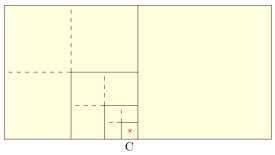


Figure 2 Let the figure show $T_{j-1} = T_1$, the true cells of a quadtree in $\mathbb{R}^2$ in white and denote the cell with the red cross as C. Cells shown with dotted lines exist but are not important for the example. Note that C is adjacent to a cell of 8 times its size so C is not 2-smooth and the parent of C is also not 2-smooth. If we want to split the neighbor of C into cells with brand 2 we create cells which are 4 times the size of C and we thus balance its parent. The second split creates cells of brand 2 which are twice the size of C and so C is 2-smooth.

Proof. Per definition a cell C is 2^j-smooth if all its neighboring leaf cells are at most a factor 2^j larger than C. This means that if we want to balance a cell C then we need to check for each of its neighboring cells if it is too large and if so, add a minimum number of cells accordingly. This makes the minimum set of cells that balances a cell C unique. If for each cell in T_j, its balancing cells are unique, then the set T_{j+1} (the union of all sets of balancing cells) is unique. ◀

▶ **Lemma 12.** *Every quadtree T^j has less than $\mathcal{O}((d2^d)^j n)$ leaf cells.*

Proof. We prove this by induction. By definition T_1 has $\mathcal{O}(n)$ cells and all cells in T_j exist to balance cells in T_{j-1}.

Now we assume that the tree T^{j-1} defined for T_1 has $\mathcal{O}((d2^d)^{j-1}n)$ leaf cells. Each of these leaf cells C can have at most d leaf neighbors which are larger than C. If such a cell C is not 2^{j-1}-smooth we need to split the too large neighboring cells. Observe that for each split we either fix the balance of C or the balance of an ancestor of C which was also in T_{j-1} and which also had to be 2^{j-1}-smooth (see Figure 2).

The result of this observation is that we perform at most d splits per leaf in T^{j-1}. Each split creates 2^d cells so T^j must have at most $d2^d\mathcal{O}((d2^d)^{j-1}n) = \mathcal{O}((d2^d)^j n)$ leaf cells.[10] ◀

▶ **Corollary 13.** *If d is constant, the tree T^* has size $\mathcal{O}(n)$.*

▶ **Lemma 14.** *Let C_1, C_2 be two family related (possibly non-leaf) cells in T^* such that $|C_1| \leq |C_2|$. Then the brand of C_2 is at most the brand of C_1.*

Proof. The proof is a proof per construction where we try to reconstruct the sequence of operations that led to the creation of cell C_1. All of the ancestors of C_1 must have a brand lower or equal to the brand of C_1, this includes the parent C_a of C_2. Since C_1 is a descendant of C_a, C_a must be split. In that split all of the children of C_a (including C_2) are created with a brand lower or equal to the brand of C_1. ◀

▶ **Lemma 15** (The Branding Principle)**.** *Let C_j be a cell in T^* with brand j. Then for any cell N neighbouring C with $|N| \geq 2^j|C_j|$, the brand of N is at most $j + 1$.*

Proof. This property follows from the definition of each set of cells T_j. If C_j is 2^j-smooth, its neighboring cells N can be at most a factor 2^j larger than C_j. When we define T_{j+1}, all

[10] We improve this bound to $\mathcal{O}(d^j 2^d n)$ leaf cells in the full version by observing that for every balance split, there are at least $\frac{1}{2}2^d$ cells demanding that the same cell must be split.

the neighbors of C_j either already have size at most $2^j|C_j|$ and thus a brand of at most j, or the neighbors must get split until they have size exactly $2^j|C_j|$. When the latter happens those cells get brand $j+1$. ◀

With these lemmas in place we are ready to prove the main result for static uncompressed quadtrees in $\mathbb{R}^1$ and $\mathbb{R}^2$.

3.2 Static uncompressed smooth quadtrees over $\mathbb{R}^1$

Let T_1 be a non-compressed quadtree over $\mathbb{R}^1$ which takes $\mathcal{O}(n)$ space. In this subsection we show that we can add at most $\mathcal{O}(n)$ cells to the quadtree T_1 such that all the cells in the resulting quadtree are 2^j-smooth for some $j \leq 2$ and the true cells are 2-smooth. Lemma 12 tells us that we can add at most $\mathcal{O}(n)$ cells with brand 2 to T_1 resulting in the tree $T^* = T_1 \cup T_2$ where all the true cells are 2-smooth in T^*.

Our claim is that in a static non-compressed quadtree over $\mathbb{R}^1$ all the cells in T_2 must be 4-smooth in T^* since we cannot have two neighboring leaf cells in T_2 with one cell more than a factor 2 larger than the other.

▶ **Theorem 16.** *Let T_1 be an uncompressed quadtree over $\mathbb{R}^1$ which takes $\mathcal{O}(n)$ space. In the smooth tree T^* there cannot be two neighboring leaf cells C_2, C_3, both with brand 2, such that $|C_2| \leq \frac{1}{2^2}|C_3|$.*[11]

Proof. The proof is by contradiction and is illustrated in Figure 3. Assume for the sake of contradiction that we have two neighboring cells C_2 and C_3 both with brand 2 with $|C_3| = 4|C_2|$. C_2 has two neighbors: one family related neighbor and one non-sibling neighbor. C_3 cannot be contained in a sibling neighbor because C_3 is larger than C_2. Note that C_2 exists to balance a true cell C_1 of smaller size. C_1 cannot be a descendant of C_3 because C_3 is a cell with brand 2. So C_1 must be a descendant of the sibling neighbor of C_2. That would make C_2 and C_1 family related and Lemma 14 then demands that C_2 has brand at most 1; a contradiction. ◀

3.3 Static uncompressed smooth quadtrees over $\mathbb{R}^2$

We also show that we can make a smooth non-compressed static quadtree over $\mathbb{R}^2$ that takes $\mathcal{O}(n)$ space, such that all the cells in the quadtree are 2^j-smooth for a $j \leq 3$. We denote the original cells by T_1 and we want them to be 2-smooth. We claim that in the extended quadtree T^* (as defined in Definition 10) all cells are 8-smooth.

[11] Careful readers can observe two things in this section: (i) Cells which are 2-smooth are allowed to have neighbors which are 4 times as large but in $\mathbb{R}^1$ they cannot. (ii) The proof of this theorem actually shows that C_3 can not even be a factor two larger than C_2. We choose not to tighten the bounds because these two observations do not generalize to higher dimensions.

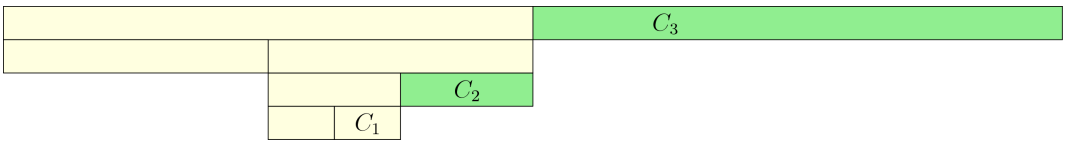


Figure 3 Two neighboring cells with brand 2 in a one-dimensional quadtree. In the figure white cells have brand 1 and light green cells have brand 2.

▶ **Theorem 17.** *Let T_1 be an uncompressed quadtree over $\mathbb{R}^2$ which takes $\mathcal{O}(n)$ space. In the extended tree T^* there cannot be two neighboring leaf cells C_3, C_4, both with brand 3, such that $|C_3| \leq \frac{1}{2^3}|C_4|$.*

Proof. The proof resembles the proof of Theorem 16, and it is illustrated in Figure 4. However, it requires two cases instead of one. Note that for C_3 to exist there must be at least two consecutive neighbors of C_3, (C_2 and C_1) with brand 2 and 1 respectively, such that $|C_1| = \frac{1}{2}|C_2| = \frac{1}{2} \cdot \frac{1}{4}|C_3|$.

Observe that C_1, C_2 and C_3 cannot be family related because of Lemma 14 and observe that C_4 can not be a sibling neighbor of C_3. The proof claims that it is impossible to place C_1, C_2, C_3 and C_4 in the plane without either violating the branding principle (Lemma 15), Lemma 14, or causing a cell with brand 1 or 2 to be not smooth.

Our first claim is that C_3 must share a vertex with C_4 (and similarly, C_2 must share a vertex with C_3). If this is not the case all the neighbors of C_3 (apart from C_4) are either contained in sibling neighbors of C_3 or neighbors of C_4. However that would imply that either C_2 is family related to C_3 or that an ancestor of C_2 of size $|C_3|$ is a neighbor of C_4. The first case cannot happen because of Lemma 14, in the second case we have a cell with brand 2 neighboring C_4 which is $\frac{1}{8}$'th the size of C_4 so C_4 must have been split but C_4 must be a leaf. Without loss of generality we say that C_3 shares the top left vertex with C_4 (Figure 4).

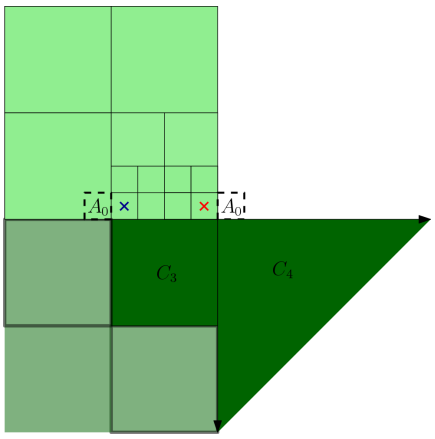


Figure 4 Two neighboring cells with brand 3 in a two-dimensional quadtree. The cells with brand 2 are light green and cells with brand 3 are dark green.

Since C_2 cannot be placed in a sibling neighbor of C_3, C_2 must be placed in the positive $\vec{y}$ direction from C_3. C_2 must also share a vertex with C_3 so we distinguish between two cases: C_2 shares the top left vertex with C_3 or the top right.

Case 1: The top left vertex. In this case C_2 is the blue square in Figure 4. C_1 cannot be contained in a sibling neighbor of C_2 so C_1 must lie to the left. However if C_1 is adjacent to C_2, its parent A_0 (the dashed lines in the figure) must also be a neighbor of C_2. Because we placed C_4 and C_3 without loss of generality, Figure 4 shows us that A_0 must neighbor a sibling neighbor of C_3 which we will denote as $F(C_3)$. We know that $|F(C_3)| = |C_3| \geq 2^1|A_0|$ and that A_0 has brand 1 and $F(C_3)$ has brand 3. This is a contradiction with the *Branding Principle* (Lemma 15).

Case 2: The top right vertex. In this case C_2 is the red square in the figure. C_1 cannot be contained in a sibling neighbor of C_2 so C_1 must lie to the right. However, if C_1 is adjacent to C_2, its parent A_0 (the dashed lines in the figure) must also be a neighbor of C_2. Moreover (for similar reasons as the first case) A_0 must also be a neighbor of C_4. We know that A_0 is the ancestor of a true cell, so A_0 has brand 1. Moreover, $|C_4| \geq 8|A_0|$ so C_4 must have been split which contradicts that C_4 is a leaf.

Both cases lead to a contradiction so Theorem 17 is proven. The structure of this proof is identical to the structure of the proof of the generalized theorem in the full version. ◀

4 Dynamic quadtrees

In the previous section we have shown that, given a static uncompressed quadtree T_1 of $\mathcal{O}(n)$ size over $\mathbb{R}^1$ or $\mathbb{R}^2$, we can create a static smooth tree T^* of $\mathcal{O}(n)$ size. In this section we prove that if T_1 is a dynamic tree, we can also dynamically maintain its extended variant T^*.

Let T_1 be a quadtree over $\mathbb{R}^1$ or $\mathbb{R}^2$ subject to the split and merge operation (Table 1 I.). If we use the split operation to create new true cells in T_1 then in T^* we (possibly) need to add cells (to the set T_2) that smooth the new true cells. Similarly if we add cells to T_2 we might need to add cells to T_3. The first question that we ask is: can we create a new split and merge operation that takes a constant number of steps per operation to maintain T^*?

▶ **Lemma 18.** *Given an uncompressed non-smooth quadtree T_1 in $\mathbb{R}^d$ of $\mathcal{O}(n)$ size and its extended tree T^*. Let T_1' be an uncompressed non-smooth quadtree such that T_1 can become T_1' with one merge or split operation. Then T^* and T'^* differ by at most $(2d)^d$ quadtree cells.*

Proof. In the proof of Lemma 12 we showed that each cell in T_j had at most d balancing cells in T_{j+1}. So if we add a new cell in T_1 with the split operation, we need to do at most d^d split operations to create the d^d cells to smooth the tree up to the level $(d+1)$.[12] d^d split operations create at most $(2d)^d$ cells.

Lemma 11 states that for each set of true cells T_1 the tree T^* is unique. So if we want to **merge** four cells in T_1 we must get a new unique T_1' and T'^*. We know that we can go from T'^* to T^* with d^d **split operations**, so we can also go from T^* to T'^* with d^d **merge** operations. ◀

4.1 The algorithm that maintains T^*

Lemma 18 tells us that it should be possible to dynamically maintain our extended quadtree T^* with $2^{\mathcal{O}(d\log(d))}$ operations per split or merge in the true tree T_1. The lemma does not specify which cells exactly need to be split. We note that our extended quadtree T^* is unique and thus independent from the order in which we split cells in T_1. In the full version we show that this property prohibits the naive implementation (just split all cells which conflict with another cell's smoothness): this does not maintain a quadtree that follows the definition of T^* given by Definition 9 (and thus does not have to be smooth). Instead we introduce the following lemma which will help us design a correct algorithm for maintaining T^*:

▶ **Lemma 19.** *Given a dynamic quadtree T_1 and its dynamic extended quadtree T^* with $T_1 \subset T^*$ where T^* is defined according to Definition 9. If a cell $C \in T^*$ has brand $j > 1$ then there is at least one neighbor N of C such that N has brand $j-1$ and $|C| = 2^{j-1}|N|$.*

Proof. If C has brand j, then according to Definition 9, C exists to smooth a cell $N \in T_{j-1}$. Per definition N has brand $j-1$ so $|C|$ is indeed $2^{j-1}|N|$. ◀

This observation allows us to devise an algorithm that maintains T^* after a split operation in T_1, we call this algorithm the **aftersplit procedure**. The first change to our static construction is that each cell $C \in T^*$ gets a collection of brands (which can contain duplicates). The current brand of a cell is the minimum of its collection of brands. Given this new definition of a brand we define a two-phased procedure:

[12] Section 5 will show that the $(d+1)$'th level is always 2^{d+1}-smooth

- Whenever we split a cell C with brand j, we check all the d neighboring *leaf* cells N that are larger than C. If N is more than a factor 2^{j-1} larger than C, the new children of C are non-smooth and N should be split into cells with brand $j+1$. If a neighbor N is split, we also invoke the aftersplit procedure on N.
- Secondly we consider this: for any neighboring leaf N of C we check there exists a cell C' with C' equal to N or an ancestor of N with the following property: C' is exactly a factor 2^{j-1} larger than C. In that case C' could exist to smooth the new children of C so we add the brand $(j+1)$ to its set of brands. We call this **rebranding**.

Algorithm 1 The procedure for after splitting a cell.

```
1: procedure AfterSplit(Cell C, Integer j)
2:     for Cell N ∈ LargerLeafNeighbors do
3:         if |N|/|C| > 2^{j-1} then
4:             V ← Split(N, j+1)
5:             AfterSplit(N, j+1)
6:         if ∃C' ∈ N ∪ Ancestors(N) such that |C'|/|C| = 2^{j-1} then
7:             C'.Brands.Add(j+1)
8:             if Changed(C'.Brands.Minimum) then
9:                 AfterSplit(C', C'.Brands.Minimum)
```

▶ **Lemma 20.** *If we split a cell with brand j, the aftersplit procedure performs at most $(2d)^{d-j}$ split and merge operations and the resulting extended quadtree T^* implements Definition 9.*

Proof. Observe that if we split a cell C with brand j, C has at most d neighbors which are at least a factor 2^{j-1} larger than C and a leaf in T^*. We can find the larger leaf neighbors with at most d level pointer traversals and the neighbors of exactly 2^{j-1} size by first finding an ancestor of C and using that ancestor's level pointers. If we **rebrand** or split one of the found neighbors, that neighbor gets a brand one greater than j so the aftersplit procedure will recurse with a new $j' = j+1$. This means that we recurse at most $d-j$ times which makes the aftersplit procedure on a cell with brand j perform d^{d-j} split operations which creates at most $(2d)^{d-j}$ cells.

The resulting tree T^* must implement Definition 9 because of Lemma 19. This lemma states that any neighboring cell N with $|N| = 2^{j-1}|C|$ could exist to smooth a child cell of C in the static scenario, so each N should contain the option to have that brand. ◀

The AfterMerge procedure is simply the inverse of the Aftersplit procedure. If we merge cells into a cell C, we check all neighbors of size $2^{j-1}|C|$ and $2^{j-2}|C|$. In the first case, we remove the brand j from the cell and check if it still has to exist. In the second case, we remove the brand j from the cell and check if it needs to be rebranded to a higher brand.

▶ **Theorem 21.** *For each dynamic compressed quadtree T_1 over $\mathbb{R}^d$ we can maintain the extended variant T^* with at most $\mathcal{O}((2d)^d)$ split and merge operations in T^* per one split or merge operation in T_1.*

5 Extending the proof of Section 3 to work for $\mathbb{R}^d$

In this section, we prove that T^* is smooth in every dimension d if T_1 is an uncompressed quadtree. Due to space restrictions, we only summarize the result here; the details can be found in the full version of the paper.

In Section 4 we already elaborated on how we can dynamically maintain our extended quadtree T^* in at most $\mathcal{O}((2d)^d)$ operations per split and merge on the true tree T_1. What remains to be proven is that the extended quadtree is indeed a smooth quadtree. We prove this by proving a generalized version of Theorem 17 in Section 3:

▶ **Theorem 22.** *Let T_1 be non-compressed quadtree over $\mathbb{R}^d$ which takes $\mathcal{O}(n)$ space. In the extended tree T^* there cannot be two neighboring leaf cells which we will name C_{d+1}, C_{d+2} with both brand $(d+1)$ such that $|C_{d+1}| \leq \frac{1}{2^{d+1}}|C_{d+2}|$.*

The proof is similar to the proof in Section 3.3. If the two neighboring cells C_{d+1} and C_{d+2} want to exist in the plane then there must be a **chain** of d cells (C_d to C_1) with decreasing brand that forces the cells to exist. We show that it is impossible to embed this chain into the plane without either needing to split the largest cell or violating the *Branding Principle* (Lemma 15). The difficulty of this proof is that there are many ways to embed such a chain of cells in $\mathbb{R}^d$. Moreover, if you want a cell C_j in the chain to cause a contradiction with a neighboring cell you need to prove that the neighbor actually exists. We define an abstract virtual operator that can traverse the extended quadtree. With that operator and Lemma 14 in place we show that if you want to place C_{d+2} to C_1 in the pane, then each time we must let C_j neighbor C_{j+1} in a unique dimension. If we use a dimension $\vec{y}$ twice we distinguish between two cases:

The first case is that we use the same dimension and same direction. This case is equal to case 1 in Section 3.3 and we show that we must violate the *Branding Principle*.

The second case is that we use the same dimension and opposite direction and it corresponds with case 2 in Section 3.3. In this case we show that C_{d+2} must have been split.

The result is that we need $(d+1)$ different dimensions to embed the chain but we have only d, a contradiction.

6 Compressed quadtrees in $\mathbb{R}^d$

The remainder of our work focuses on building smooth quadtrees that store a set of points and that are dynamic with respect to insertions and deletions of points from the set. If we want to build a dynamic quadtree over a point set of unbounded spread we need the quadtree to be compressed. We again want to define an extended quadtree T^* that we can maintain with the operations in Section 4. In this section summarize the extention of our results to compressed quadtrees. The details can be found in the full version of the paper.

Recall that in Definition 10 we had $(d+1)$ different sets of cells where each set T_j was defined as the minimal number of cells needed to balance the cells in T_{j-1}. We showed that if we have two uncompressed quadtrees T_1 and T_1' which only differ by one split/merge operation that their extended quadtrees T^* and T'^* only differ in $\mathcal{O}((2d)^d)$ cells. However, if T_1 and T_1' are uncompressed quadtrees and if we use Definition 9, the reader can imagine that there are scenarios where their extended quadtrees T^* and T'^* differ in an arbitrarily large number of cells (see the full version). This is why instead we redefine our extended quadtree as the union of all extended quadtrees A^* of all uncompressed components A_1 of T_1: $T^* := \cup_{Uncompressed\ components A_1} A^*$. If we only demand that cells in uncompressed components count towards smoothness (the other cells are not stored anyway) then this extended quadtree is clearly smooth if Theorem 22 holds. However, Theorem 22 relies on Lemma 14 that says that cells that are *family related* have a related brand and with this definition of T^* we can place cells with an arbitrary high brand "on top" of other cells. We observe that the proof of Theorem 22 only uses paths that traverse d^2 levels of depth in the

tree and we define additional operators that make sure that for each cell C, all ancestors within d^2 levels of C have a correct brand. The result is the following theorem:

▶ **Theorem 23.** *For each dynamic compressed quadtree T_1 over $\mathbb{R}^d$ we can maintain a smooth variant T^* with at most:*

- $\mathcal{O}((2d)^d)$ *operations per split or merge on T_1.*
- $\mathcal{O}(\log(\alpha)d^2(6d)^{d+1})$ *operations per deletion or insertion of a compressed leaf.*
- $\mathcal{O}(d^2(6d)^d)$ *time per upgrowing or downgrowing.*

7 Alignment in Real RAM

Finally, we show that we can maintain non-aligned compressed quadtrees. Here we only give a summary of this part; for the details see the full version.

In the previous section we were treating our compressed quadtree as if for each compressed component A_1, the cell of its root R was *aligned* with the cell of the leaf C_a that stores its compressed link. That is, the cell of R can be obtained by repeatedly subdividing the cell of C_a. However, finding an aligned cell for the root of a new uncompressed component when it is inserted is not supported in constant time in Real RAM model of computation. The reason is that the length of the compressed link, and thus the number of divisions necessary to compute the aligned root cell, may be arbitrary.

Instead of computing the aligned root cell at the insertion of each new compressed component, we allow compressed nodes to be associated with any hypercube of an appropriate size that is contained in the cell of the compressed ancestor C_a. While doing so, we ensure that the following **alignment property** is maintained: For any compressed link in T^*, if the length of the link is at most 4α, then the corresponding uncompressed component is aligned with the parent cell of the link.

In the full version of [19], Löffler *et al.* obtain this in amortized additional $O(1)$ time by relocating the uncompressed component just before the decompression operation starts. To accomplish the relocation task, they exploit the algorithm of Löffler and Mulzer [18] that, given a compressed quadtree T in $\mathbb{R}^2$ and an appropriate square S, produces a 2-smooth compressed quadtree T' on S that stores the same point set as T. We modify the result of [19] in three ways: (i) We de-amortize it by adapting a well-known de-amortization technique that gradually constructs the correctly aligned copy of an unaligned uncompressed component [16]. (ii) We adapt the algorithm of [18] so that it produces our target quadtree T^* instead of a 2-smooth compressed quadtree. (iii) By extending the analysis in [18], we show that the algorithm to produce T^* works in $\mathbb{R}^d$, and its time complexity is linear in the size of the relocated tree.

The main result of this section is as follows.

▶ **Theorem 24.** *Let P be a point set in $\mathbb{R}^d$ and T_1 be an α-compressed quadtree over P for a sufficiently big constant α with $\alpha > 2^{2^d}$, and let T^* be the extended variant of a compressed quadtree T_1 with the definition in [14]. The operations split and upgrowing on T^* can be modified to maintain the alignment property for T^*. The modified operations require worst-case $O(32^d + d^2 \cdot (6d)^d)$ time.*

References

1 Boris Aronov, Hervé Brönnimann, Allen Y Chang, and Yi-Jen Chiang. Cost prediction for ray shooting in octrees. *Computational Geometry*, 34(3):159–181, 2006.

2 Huck Bennett, Evanthia Papadopoulou, and Chee Yap. Planar minimization diagrams via subdivision with applications to anisotropic Voronoi diagrams. *Comput. Graph. Forum*, 35(5):229–247, 2016.

3 Huck Bennett and Chee Yap. Amortized analysis of smooth quadtrees in all dimensions. *Computational Geometry*, 63:20–39, 2017.

4 Marshall Bern, David Eppstein, and John Gilbert. Provably good mesh generation. *Journal on Computational System Sciences*, 48(3):384–409, 1994.

5 Marshall Bern, David Eppstein, and Shang-Hua Teng. Parallel construction of quadtrees and quality triangulations. *International Journal on Computational Geometry and Applications*, 9(6):517–532, 1999.

6 Kevin Buchin and Wolfgang Mulzer. Delaunay triangulations in $O(\texttt{sort}(n))$ time and more. *Journal of the ACM*, 58(2):Art. 6, 2011.

7 Paul B. Callahan and S. Rao Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *Journal of the ACM*, 42(1):67–90, 1995.

8 Mark De Berg, Otfried Cheong, Marc Van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 2008.

9 Mark De Berg, Marcel Roeloffzen, and Bettina Speckmann. Kinetic compressed quadtrees in the black-box model with applications to collision detection for low-density scenes. *Algorithms–ESA 2012*, pages 383–394, 2012.

10 Olivier Devillers, Stefan Meiser, and Monique Teillaud. Fully dynamic delaunay triangulation in logarithmic expected time per operation. *Computational Geometry*, 2(2):55–80, 1992.

11 David Eppstein, Michael Goodrich, and Jonathan Sun. The skip quadtree: a simple dynamic data structure for multidimensional data. *International Journal on Computational Geometry and Applications*, 18(1–2):131–160, 2008.

12 Raphael A. Finkel and Jon Louis Bentley. Quad trees: A data structure for retrieval on composite keys. *Acta Inform.*, 4:1–9, 1974.

13 Sariel Har-Peled. *Geometric approximation algorithms*, volume 173. American mathematical society Boston, 2011.

14 Ivor Hoog v.d., Elena Khramtcova, and Maarten Löffler. Dynamic smooth compressed quadtrees. *arXiv preprint arXiv:1712.05591*, 2017.

15 Elena Khramtcova and Maarten Löffler. Dynamic stabbing queries with sub-logarithmic local updates for overlapping intervals. In *Proc. 12th Int. Computer Science Symp. in Russia, (CSR)*, pages 176–190, 2017.

16 S Rao Kosaraju and Mihai Pop. De-amortization of algorithms. In *International Computing and Combinatorics Conference*, pages 4–14. Springer, 1998.

17 Drago Krznaric and Christos Levcopoulos. Computing a threaded quadtree from the Delaunay triangulation in linear time. *Nordic Journal on Computating*, 5(1):1–18, 1998.

18 Maarten Löffler and Wolfgang Mulzer. Triangulating the square and squaring the triangle: quadtrees and Delaunay triangulations are equivalent. *SIAM Journal on Computing*, 41(4):941–974, 2012.

19 Maarten Löffler, Joseph A. Simons, and Darren Strash. Dynamic planar point location with sub-logarithmic local updates. In *13th Int. Symp. Algorithms and Data Structures (WADS)*, pages 499–511. Springer Berlin Heidelberg, 2013. full version: arXiv preprint arXiv:1204.4714.

20 J David MacDonald and Kellogg S Booth. Heuristics for ray tracing using space subdivision. *The Visual Computer*, 6(3):153–166, 1990.

21 Johannes Mezger, Stefan Kimmerle, and Olaf Etzmuß. Hierarchical techniques in collision detection for cloth animation. In *Proc. Int. Conf. in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*, 2003.

22 Eunhui Park and David Mount. A self-adjusting data structure for multidimensional point sets. *Algorithms–ESA 2012*, pages 778–789, 2012.

23 Hanan Samet. Neighbor finding techniques for images represented by quadtrees. *Computer graphics and image processing*, 18(1):37–57, 1982.

24 Hanan Samet. *The design and analysis of spatial data structures*. Addison-Wesley, Boston, MA, USA, 1990.

25 A Unnikrishnan, Priti Shankar, and YV Venkatesh. Threaded linear hierarchical quadtrees for computation of geometric properties of binary images. *IEEE transactions on software engineering*, 14(5):659–665, 1988.

26 Pravin M. Vaidya. Minimum spanning trees in k-dimensional space. *SIAM J. Comput.*, 17(3):572–582, 1988. doi:10.1137/0217035.

On the Treewidth of Triangulated 3-Manifolds

Kristóf Huszár
Institute of Science and Technology Austria (IST Austria)
Am Campus 1, 3400 Klosterneuburg, Austria
kristof.huszar@ist.ac.at
https://orcid.org/0000-0002-5445-5057

Jonathan Spreer[1]
Institut für Mathematik, Freie Universität Berlin
Arnimallee 2, 14195 Berlin, Germany
jonathan.spreer@fu-berlin.de
https://orcid.org/0000-0001-6865-9483

Uli Wagner
Institute of Science and Technology Austria (IST Austria)
Am Campus 1, 3400 Klosterneuburg, Austria
uli@ist.ac.at
https://orcid.org/0000-0002-1494-0568

Abstract

In graph theory, as well as in 3-manifold topology, there exist several width-type parameters to describe how "simple" or "thin" a given graph or 3-manifold is. These parameters, such as pathwidth or treewidth for graphs, or the concept of thin position for 3-manifolds, play an important role when studying algorithmic problems; in particular, there is a variety of problems in computational 3-manifold topology – some of them known to be computationally hard in general – that become solvable in polynomial time as soon as the dual graph of the input triangulation has bounded treewidth.

In view of these algorithmic results, it is natural to ask whether every 3-manifold admits a triangulation of bounded treewidth. We show that this is not the case, i.e., that there exists an infinite family of closed 3-manifolds not admitting triangulations of bounded pathwidth or treewidth (the latter implies the former, but we present two separate proofs).

We derive these results from work of Agol and of Scharlemann and Thompson, by exhibiting explicit connections between the topology of a 3-manifold $\mathcal{M}$ on the one hand and width-type parameters of the dual graphs of triangulations of $\mathcal{M}$ on the other hand, answering a question that had been raised repeatedly by researchers in computational 3-manifold topology. In particular, we show that if a closed, orientable, irreducible, non-Haken 3-manifold $\mathcal{M}$ has a triangulation of treewidth (resp. pathwidth) k then the Heegaard genus of $\mathcal{M}$ is at most $48(k+1)$ (resp. $4(3k+1)$).

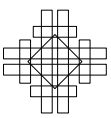
2012 ACM Subject Classification Mathematics of computing → Geometric topology, Theory of computation → Fixed parameter tractability

Keywords and phrases computational topology, triangulations of 3-manifolds, thin position, fixed-parameter tractability, congestion, treewidth

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.46

Related Version A full version of the paper is available at [26], `http://arxiv.org/abs/1712.00434`.

[1] Research of the second author was supported by the Einstein Foundation (project "Einstein Visiting Fellow Santos") and by the Simons Foundation ("Simons Visiting Professors" program).



Acknowledgements We would like to thank Arnaud de Mesmay and Stephan Tillmann for helpful discussions and remarks, and the anonymous reviewers for useful comments and suggestions regarding the exposition. This work was initiated during a visit of the second author at IST Austria. The second author would like to thank the people at IST Austria for their hospitality.

1 Introduction

In the field of 3-manifold topology many fundamental problems can be solved algorithmically. Famous examples include deciding whether a given knot is trivial [21], deciding whether a given 3-manifold is homeomorphic to the 3-sphere [40, 47], and, more generally (based on Perelman's proof of Thurston's geometrization conjecture [29]), deciding whether two given 3-manifolds are homeomorphic, see, e.g., [2, 31, 45]. The algorithm for solving the homeomorphism problem is still purely theoretical, and its complexity remains largely unknown [31, 33]. In contrast, the first two problems are known to lie in the intersection of the complexity classes **NP** and **co-NP** [22, 27, 30, 32, 43, 49].[2]

Moreover, implementations of, for instance, algorithms to recognize the 3-sphere exist out-of-the-box (e.g., using the computational 3-manifold software *Regina* [9]) and exhibit practical running times for virtually all known inputs.

In fact, many topological problems with implemented algorithmic solutions solve problem instances of considerable size. This is despite the fact that most of these implementations have prohibitive worst-case running times, or the underlying problems are even known to be computationally hard in general. In recent years, there have been several attempts to explain this gap using the concepts of parameterized complexity and algorithms for fixed parameter tractable (FPT) problems [19]. This effort has proven to be highly effective and, today, there exist numerous FPT algorithms in the field [10, 12, 13, 14, 34]. More specifically, given a triangulation $\mathcal{T}$ of a 3-manifold $\mathcal{M}$ with n tetrahedra whose dual graph $\Gamma(\mathcal{T})$ has treewidth[3] at most k, there exist algorithms to compute

- taut angle structures[4] of what is called ideal triangulations with torus boundary components in running time $O(7^k \cdot n)$ [14];
- optimal Morse matchings[5] in the Hasse diagram of $\mathcal{T}$ in $O(4^{k^2+k} \cdot k^3 \cdot \log k \cdot n)$ [12];
- the Turaev–Viro invariants[6] for parameter $r \geq 3$ in $O((r-1)^{6(k+1)} \cdot k^2 \cdot \log r \cdot n)$ [13];
- every problem which can be expressed in monadic second-order logic in $O(f(k) \cdot n)$, where f often is a tower of exponentials [10].[7]

Some of these results are not purely theoretical – as is sometimes the case with FPT algorithms – but are implemented and outperform previous state-of-the-art implementations for typical input. As a result, they have a significant practical impact. This is in particular the case for the algorithm to compute Turaev–Viro invariants [13, 34].

Note that treewidth – the dominating factor in the running times given above – is a combinatorial quantity linked to a triangulation, not a topological invariant of the underlying

[2] The proof of **co-NP** membership for 3-sphere recognition assumes the Generalized Riemann Hypothesis.
[3] We often simply speak of the treewidth of a triangulation, meaning the treewidth of its dual graph.
[4] Taut angle structures are combinatorial versions of semi-simplicial metrics which have implications on the geometric properties of the underlying manifold.
[5] Optimal Morse matchings translate to discrete Morse functions with the minimum number of critical points with respect to the combinatorics of the triangulation and the topology of the underlying 3-manifold.
[6] Turaev–Viro invariants are powerful tools to distinguish between 3-manifolds. They are the method of choice when, for instance, creating large censuses of manifolds.
[7] This result is analogous to Courcelle's celebrated theorem in graph theory [16].

manifold. This gives rise to the following approach to efficiently solve topological problems on a 3-manifold $\mathcal{M}$: given a triangulation $\mathcal{T}$ of $\mathcal{M}$, search for a triangulation $\mathcal{T}'$ of the same manifold with smaller treewidth.

This approach faces severe difficulties. By a theorem due to Kirby and Melvin [28], the Turaev–Viro invariant for parameter $r = 4$ is #**P**-hard to compute. Thus, if there were a polynomial time procedure to turn an n-tetrahedron triangulation $\mathcal{T}$ into a poly(n)-tetrahedron triangulation $\mathcal{T}'$ with dual graph of treewidth at most k, for some universal constant k, then this procedure, combined with the algorithm from [13], would constitute a polynomial time solution for a #**P**-hard problem. Furthermore, known facts imply that *most* triangulations of *most* 3-manifolds must have large treewidth.[8] However, while these arguments indicate that triangulations of small treewidth may be rare and computationally hard to find, it does not rule out that every manifold has some (potentially very large) triangulation of bounded treewidth.

In this article we show that this is actually not the case, answering a question that had been raised repeatedly by researchers in computational 3-manifold topology.[9] More specifically, we prove the following two statements.

▶ **Theorem 1.** *There exists an infinite family of* 3*-manifolds which does not admit triangulations with dual graphs of uniformly bounded pathwidth.*

▶ **Theorem 2.** *There exists an infinite family of* 3*-manifolds which does not admit triangulations with dual graphs of uniformly bounded treewidth.*

We establish the above results through the following theorems, which are the main contributions of the present paper. The necessary terminology is introduced in Section 2.

▶ **Theorem 3.** *Let $\mathcal{M}$ be a closed, orientable, irreducible, non-Haken* 3*-manifold and let $\mathcal{T}$ be a triangulation of $\mathcal{M}$ with dual graph $\Gamma(\mathcal{T})$ of pathwidth $\mathrm{pw}(\Gamma(\mathcal{T})) \leq k$. Then $\mathcal{M}$ has Heegaard genus $\mathfrak{g}(\mathcal{M}) \leq 4(3k+1)$.*

▶ **Theorem 4.** *Let $\mathcal{M}$ be a closed, orientable, irreducible, non-Haken* 3*-manifold and let $\mathcal{T}$ be a triangulation of $\mathcal{M}$ with dual graph $\Gamma(\mathcal{T})$ of treewidth $\mathrm{tw}(\Gamma(\mathcal{T})) \leq k$. Then $\mathcal{M}$ has Heegaard genus $\mathfrak{g}(\mathcal{M}) < 48(k+1)$.*

By a result of Agol [1] (Theorem 12 in this paper), there exist closed, orientable, irreducible, non-Haken 3-manifolds of arbitrarily large Heegaard genus. Combining this result with Theorems 3 and 4 thus immediately implies Theorems 1 and 2.

▶ Remark. Note that Theorem 1 can be directly deduced from Theorem 2 since the pathwidth of a graph is always at least as large as its treewidth.[10] Nonetheless, we provide separate proofs for each of the two statements. The motivation for this is that all the ingredients to prove Theorem 3 (and hence Theorem 1) already appear in the literature, while Theorem 4 needs extra work to be done.

[8] It is known that, given $k \in \mathbb{N}$, there exist constants $C, C_k > 1$ such that there are at least $C^{n\log(n)}$ 3-manifolds which can be triangulated with $\leq n$ tetrahedra, whereas there are at most C_k^n triangulations with treewidth $\leq k$ and $\leq n$ tetrahedra. See the full version of the article for more details.

[9] The question whether every 3-manifold admits a triangulation of bounded treewidth, and variations thereof have been asked at several meetings and open problem sessions including an Oberwolfach meeting in 2015 [11, Problem 8] (formulated in the context of knot theory).

[10] This is immediate from the definitions of treewidth and pathwidth, see Section 3.

The paper is organized as follows. After going over the preliminaries in Section 2, we give an overview of selected width-type graph parameters in Section 3. Most notably, we propose the *congestion* of a graph (also known as *carving width*) as an alternative choice of a parameter for FPT algorithms in 3-manifold topology. Section 4 is devoted to results from 3-manifold topology which we build upon. In Section 5 we then prove Theorem 1, and in Section 6 we prove Theorem 2.

2 Preliminaries

In this section we recall some basic concepts and terminology of graph theory, 3-manifolds, triangulations, and parameterized complexity theory.

Graphs vs. triangulations. Following several authors in the field, we use the terms *edge* and *vertex* to refer to an edge or vertex in a 3-manifold triangulation, whereas the terms *arc* and *node* denote an edge or vertex in a graph, respectively.

2.1 Graphs

For general background on graph theory we refer to [17].

A *graph* (more specifically, a *multigraph*) $G = (V, E)$ is an ordered pair consisting of a finite set $V = V(G)$ of *nodes* and a multiset $E = E(G)$ of unordered pairs of nodes, called *arcs*. We allow *loops*, i.e., an arc $e \in E$ might itself be a multiset, e.g., $e = \{v, v\}$ for some $v \in V$. The *degree* of a node $v \in V$, denoted by $\deg(v)$, equals the number of arcs containing it, counted with multiplicity. In particular, a loop $\{v, v\}$ contributes two to the degree of v. For every node $v \in V$ of a graph G, its *star* $\mathrm{st}_G(v)$ denotes the set of edges incident to v. A graph is called *k-regular* if all of its nodes have the same degree $k \in \mathbb{N}$.

2.2 3-Manifolds and their triangulations

For an introduction to the topology and geometry of 3-manifolds and to their triangulations we refer to the textbook [44] and to the seminal monograph [48].

A 3-*manifold with boundary* is a topological space[11] $\mathcal{M}$ such that each point $x \in \mathcal{M}$ has a neighborhood which either looks like (i.e., is homeomorphic to) the Euclidean 3-space $\mathbb{R}^3$ or the closed upper half-space $\{(x, y, z) \in \mathbb{R}^3 : z \geq 0\}$. The points of $\mathcal{M}$ that do not have a neighborhood homeomorphic to $\mathbb{R}^3$ constitute the *boundary* $\partial\mathcal{M}$ of $\mathcal{M}$. A 3-manifold is *bounded* (resp. *closed*) if it has a non-empty (resp. empty) boundary.

Informally, two 3-manifolds are equivalent (or *homeomorphic*) if one can be turned into the other by a continuous, reversible deformation. In other words, when talking about a 3-manifold, we are not interested in its particular shape, but only in its qualitative properties, called *topological invariants*, such as "number of boundary components", or "connectedness".

All 3-manifolds considered in this article are assumed to be compact and connected.

Handle decompositions. Every compact 3-manifold can be built from finitely many building blocks called 0-, 1-, 2-, and 3-handles. In such a *handle decomposition* all handles are (homeomorphic to) 3-balls, and are only distinguished in how they are glued to the existing decomposition.[12] For instance, to build a closed 3-manifold from handles, we may start with

[11] More precisely, we only consider topological spaces which are second countable and Hausdorff.
[12] See Chapter 6 and Appendix B of [44] for a primer on handle decompositions and Morse functions.

a collection of disjoint 3-balls, or 0-*handles*, where further 3-balls are glued to the boundary of the existing decomposition along pairs of 2-dimensional disks, the so-called 1-*handles*, or along annuli, the so-called 2-*handles*. This process is iterated until the boundary of the decomposition consists of a collection of 2-spheres. These are then eliminated by gluing in one additional 3-ball per boundary component, the 3-*handles* of the decomposition.

In every step of building up a (closed) 3-manifold from handles, the existing decomposition is a bounded 3-manifold and its boundary – called a *bounding surface* – separates the 3-manifold into two pieces: the part that is already present, and its complement (each of them possibly disconnected).

Bounding surfaces and, more generally, all kinds of surfaces embedded in a 3-manifold, play an important role in the study of 3-manifolds (similar to that of simple closed curves in the study of surfaces). When chosen carefully, an embedded surface reveals valuable information about the topology of the ambient 3-manifold. Of course, in this sense, some embedded surfaces can be considered topologically more meaningful than others. In particular, we have the following notions.

Let $\mathcal{M}$ be a 3-manifold. A surface $\mathcal{S} \subset \mathcal{M}$ is said to be *properly embedded*, if it is embedded in $\mathcal{M}$, and we have for the boundary $\partial\mathcal{S} = \mathcal{S} \cap \partial\mathcal{M}$. Let $\mathcal{S} \subset \mathcal{M}$ be a properly embedded surface distinct from the 2-sphere, and let D be a disk embedded into $\mathcal{M}$ such that its boundary satisfies $\partial D = D \cap \mathcal{S}$. D is said to be a *compressing disk for* $\mathcal{S}$ if ∂D does not bound a disk on $\mathcal{S}$. If such a compressing disk exists, then $\mathcal{S}$ is called *compressible*, otherwise it is called *incompressible*. An embedded 2-sphere $\mathcal{S} \subset \mathcal{M}$ is called *incompressible* if $\mathcal{S}$ does not bound a 3-ball in $\mathcal{M}$.[13]

A 3-manifold $\mathcal{M}$ is called *irreducible*, if every embedded 2-sphere bounds a 3-ball in $\mathcal{M}$.[14] Moreover, it is called P^2*-irreducible*, if it does not contain an embedded 2-sided[15] real projective plane $\mathbb{R}P^2$. This notion is only significant for non-orientable manifolds, since orientable 3-manifolds cannot contain any 2-sided non-orientable surfaces. If a P^2-irreducible, irreducible 3-manifold $\mathcal{M}$ contains a 2-sided incompressible surface, then it is called *Haken*. Otherwise it is called *non-Haken*. In this article, we only apply the notion of a non-Haken manifold to manifolds that are (P^2-)irreducible. In the literature, such manifolds are also referred to as *small*.

Handlebodies and compression bodies. Above we already discussed handle decompositions of 3-manifolds. Closely related are the notions of handlebody and compression body.

A *handlebody* is a bounded 3-manifold which can be described as a single 0-handle with a number of 1-handles attached to it, or, equivalently, as a thickened version of a graph.

Let $\mathcal{S}$ be a closed (not necessarily connected) surface. A *compression body* is a 3-manifold $\mathcal{N}$ obtained from $\mathcal{S} \times [0,1]$, by attaching 1-handles to $\mathcal{S} \times \{1\}$, and filling in some of the 2-sphere components of $\mathcal{S} \times \{0\}$ with 3-balls. $\mathcal{N}$ has two sets of boundary components: $\partial_-\mathcal{N} = \mathcal{S} \times \{0\} \setminus \{\text{filled in 2-sphere components}\}$ and $\partial_+\mathcal{N} = \partial\mathcal{N} \setminus \partial_-\mathcal{N}$.[16]

Dual to this construction, we can also start with $\mathcal{S} \times [0,1]$, add 2-handles along $\mathcal{S} \times \{1\}$ and fill in some resulting 2-sphere boundary components with 3-balls. Again, $\mathcal{N}$ has two sets of boundary components: $\partial_-\mathcal{N} = \mathcal{S} \times \{0\}$ and $\partial_+\mathcal{N} = \partial\mathcal{N} \setminus \partial_-\mathcal{N}$.

[13] A standard example of a compressible surface is a torus (or any other orientable surface) embedded in the 3-sphere $\mathbb{S}^3$, and of an incompressible surface is the 2-sphere $\mathbb{S}^2 \times \{x\} \subset \mathbb{S}^2 \times \mathbb{S}^1$.
[14] In particular, $\mathbb{S}^2 \times \mathbb{S}^1$ is not irreducible.
[15] A properly embedded surface $\mathcal{S} \subset \mathcal{M}$ is 2-sided in $\mathcal{M}$, if the codimension zero submanifold in $\mathcal{M}$ obtained by thickening $\mathcal{S}$ has two boundary components, i.e., $\mathcal{S}$ locally separates $\mathcal{M}$ into two pieces.
[16] If $\mathcal{S}$ is a 2-sphere, and $\mathcal{S} \times \{0\}$ is filled in, i.e., $\partial_-\mathcal{N} = \emptyset$, then $\mathcal{N}$ is actually a handlebody.

Handlebodies very naturally occur when building up a (connected) 3-manifold $\mathcal{M}$ from an arbitrary but fixed set of handles. If (possibly after deforming the attaching maps) all 0- and 1-handles are attached before attaching any 2- or 3-handles, we obtain a decomposition of $\mathcal{M}$ into two handlebodies: the union of all 0- and 1-handles on one side, and its complement on the other side. Such a decomposition exists for every 3-manifold $\mathcal{M}$ [35] and is called a *Heegaard splitting of* $\mathcal{M}$. The genus of their (common) boundary surface is called the *genus of the decomposition*. It is equal to #1-handles minus #0-handles plus one. The smallest genus of a handle decomposition of $\mathcal{M}$, which is a topological invariant by definition, is called the *Heegaard genus of* $\mathcal{M}$ and is denoted by $\mathfrak{g}(\mathcal{M})$.

Compression bodies are central to Scharlemann and Thompson's definition of thin position [42], discussed in Section 4, as they can be used to describe more complicated sequences of handle attachments, e.g., when building up a manifold by first only attaching some of the 0- and 1-handles before attaching 2- and 3-handles.

Given a possibly bounded 3-manifold $\mathcal{M}$, a fixed set of handles, and an arbitrary sequence of handle attachments to build up $\mathcal{M}$, we look at the first terms of the sequence up to (but not including) the first 2- or 3-handle attachment. Let $\mathcal{S}$ be the surface given by the boundaries of all 0-handles in this subsequence plus potentially some of the boundary components of $\mathcal{M}$ (we want the boundary components of $\mathcal{M}$ to appear at the beginning or the end of this construction). We thicken $\mathcal{S}$ into a bounded 3-manifold $\mathcal{S} \times [0,1]$, fill back in the 0-handles at the 2-sphere components of $\mathcal{S} \times \{0\}$ and attach the 1-handles from the subsequence to $\mathcal{S} \times \{1\}$. This is a compression body, say $\mathcal{N}_1$ (if no boundary components are added, $\mathcal{N}_1$ is merely the union of all 0- and 1-handles in the subsequence, and hence a union of handlebodies). In the second step we look at all 2- and 3-handles following the initial sequence of 0- and 1-handles until we reach 0- or 1-handles again, and follow the dual construction to obtain another compression body $\mathcal{K}_1$. Iterating this procedure, $\mathcal{M}$ can be decomposed into a sequence of compression bodies $(\mathcal{N}_1, \mathcal{K}_1, \mathcal{N}_2, \mathcal{K}_2, \ldots, \mathcal{N}_s, \mathcal{K}_s)$ satisfying $\mathcal{R}_i := \partial_+ \mathcal{N}_i = \partial_- \mathcal{K}_i$, $1 \leq i \leq s$, and $\mathcal{F}_i := \partial_+ \mathcal{K}_i = \partial_- \mathcal{N}_{i+1}$, $1 \leq i < s$. By construction, the surfaces $\mathcal{R}_i$ have more handles and/or less connected components than the surfaces $\mathcal{F}_i$.

When talking about bounding surfaces in sequences of handle attachments, surfaces of type $\mathcal{R}_i$ and $\mathcal{F}_i$ are typically the most interesting ones as they form local extrema with respect to their topological complexity.[17] Thus, we refer to the $\mathcal{R}_i$ as the *large* and to the $\mathcal{F}_i$ as the *small* bounding surfaces.

Triangulations. In this article we typically describe 3-manifolds by *semi-simplicial triangulations* (also known as *singular triangulations*, here often just referred to as *triangulations*). That is, a collection of n abstract tetrahedra, glued together in pairs along their triangular faces (called *triangles*). As a result of these *face gluings*, many tetrahedral edges (or vertices) are glued together and we refer to the result as a single *edge (or vertex) of the triangulation*.

A triangulation $\mathcal{T}$ describes a closed 3-manifold[18] if no tetrahedral edge is identified with itself in reverse, and the boundary of a small neighborhood around each vertex is a 2-sphere.

Given a triangulation $\mathcal{T}$ of a closed 3-manifold, its *dual graph* $\Gamma(\mathcal{T})$ (also called the *face pairing graph*) is the graph with one node per tetrahedron of $\mathcal{T}$, and with an arc between two nodes for each face gluing between the corresponding pair of tetrahedra. By construction, the dual graph is a 4-regular multigraph. Since every triangulation $\mathcal{T}$ can be linked to its dual graph $\Gamma(\mathcal{T})$ this way, we often attribute properties of $\Gamma(\mathcal{T})$ directly to $\mathcal{T}$.

[17] We specify precisely what we mean by the *topological complexity of a surface* whenever this is necessary.
[18] It is straightforward to extend the definition of a triangulation to include bounded 3-manifolds. However, for the purpose of this article it suffices to consider the closed case.

2.3 Parameterized complexity and fixed parameter tractability

There exist various notions and concepts of a refined complexity analysis for theoretically difficult problems. One of them, parameterized complexity, due to Downey and Fellows [18, 19], identifies a parameter on the set of inputs, which is responsible for the hardness of a given problem.

More precisely, for a problem $\mathcal{P}$ with input set $\mathcal{A}$, a parameter is a (computable) function $p\colon \mathcal{A} \to \mathbb{N}$. If the parameter p is the output of $\mathcal{P}$, then p is called the *natural parameter*. The problem $\mathcal{P}$ is said to be *fixed parameter tractable for parameter* p (or *FPT in* p for short) if there exists an algorithm which solves $\mathcal{P}$ for every instance $A \in \mathcal{A}$ with running time $O(f(p(A)) \cdot \mathrm{poly}(n))$, where n is the size of the input A, and $f\colon \mathbb{N} \to \mathbb{N}$ is arbitrary (computable). By definition, such an algorithm then runs in polynomial time on the set of inputs with bounded p. Hence, this identifies, in some sense, p as a potential "source of hardness" for $\mathcal{P}$ (cf. the results listed in the Introduction).

In computational 3-manifold topology, a very important set of parameters is the one of topological invariants, i.e., properties which only depend on the topology of a given manifold and are independent of the choice of triangulation (see [34] for such a result, using the first Betti number as parameter). However, most FPT-results in the field use parameters of the dual graph of a triangulation which greatly depend on the choice of the triangulation: every 3-manifold admits a triangulation with arbitrarily high graph parameters – for all parameters considered in this article.

The aim of this work is to link these parameters to topological invariants in the only remaining possible sense: given a 3-manifold $\mathcal{M}$, find lower bounds for graph parameters of dual graphs of triangulations ranging over all triangulations of $\mathcal{M}$.

3 Width-type graph parameters

The theory of parameterized complexity has its sources in graph theory, where many problems which are **NP**-hard in general become tractable in polynomial time if one assumes structural restrictions about the possible input graphs.

For instance, several graph theoretical questions have a simple answer if one asks them about trees, or graphs which are similar to trees in some sense. Width-type parameters make this sense of similarity precise [24]. We are particularly interested in the behavior of these parameters and their relationship with each other when considering bounded-degree graphs or, more specifically, dual graphs of 3-manifold triangulations.

Treewidth and pathwidth. The concepts of treewidth and pathwidth were introduced by Robertson and Seymour in their early papers on graph minors [38, 39], also see the surveys [5, 7, 8]. Given a graph G, its *treewidth* $\mathrm{tw}(G)$ measures how tree-like the graph is.

▶ **Definition 5** (Tree decomposition, treewidth)**.** A *tree decomposition* of a graph $G = (V, E)$ is a tree T with nodes $B_1, \ldots, B_m \subseteq V$, also called *bags*, such that

1. $B_1 \cup \ldots \cup B_m = V$,
2. if $v \in B_i \cap B_j$ then $v \in B_k$ for all bags B_k of T in the path between B_i and B_j, in other words, the bags containing v span a (connected) subtree of T,
3. for every arc $\{u, v\} \in E$, there exists a node B_i such that $\{u, v\} \subseteq B_i$.

The *width* of a tree decomposition equals the size of the largest bag minus one. The *treewidth* $\mathrm{tw}(G)$ is the minimum width among all possible tree decompositions of G.

▶ **Definition 6** (Path decomposition, pathwidth)**.** A *path decomposition* of a graph $G = (V, E)$ is a tree decomposition for which the tree T is required to be a *path*. The *pathwidth* $\mathrm{pw}(G)$ of a graph G is the minimum width of any path decomposition of G.

Cutwidth. The *cutwidth* $\mathrm{cw}(G)$ of a graph G is the graph-analogue of the linear width of a manifold (to be discussed in Section 4). If we order the nodes $\{v_1, \ldots, v_n\} = V(G)$ of G on a line, the set of arcs running from a node v_i, $i \leq \ell$, to a node v_j, $j > \ell$, is called a cutset C_ℓ of the ordering. The cutwidth $\mathrm{cw}(G)$ is defined to be the cardinality of the largest cutset, minimized over all linear orderings of $V(G)$.

Cutwidth and pathwidth are closely related: for bounded-degree graphs they are within a constant factor. Let $\Delta(G)$ denote the maximum degree of a node in G.

▶ **Theorem 7** (Bodlaender, Theorems 47 and 49 from [6][19])**.** *Given a graph G, we have*

$$\mathrm{pw}(G) \leq \mathrm{cw}(G) \leq \Delta(G)\,\mathrm{pw}(G).$$

Congestion. Bienstock introduced *congestion* [4], a generalization of cutwidth, which is a quantity related to treewidth in a similar way as cutwidth to pathwidth (compare Theorems 7 and 9).

Let us consider two graphs G and H, called the *guest* and the *host*, respectively. An *embedding* $\mathcal{E} = (\iota, \rho)$ of G into H consists of an injective mapping $\iota\colon V(G) \to V(H)$ together with a routing ρ that assigns to each arc $\{u, v\} \in E(G)$ a path in H with endpoints $\iota(u)$ and $\iota(v)$. If $e \in E(G)$ and $h \in E(H)$ is on the path $\rho(e)$, then we say that "e is running parallel to h". The congestion of G with respect to an embedding $\mathcal{E}$ of G into a host graph H, denoted as $\mathrm{cng}_{H,\mathcal{E}}(G)$, is defined to be the maximal number of times an arc of H is used in the routing of arcs of G.

Several notions of congestion can be obtained by minimizing $\mathrm{cng}_{H,\mathcal{E}}(G)$ over various families of host graphs and embeddings (see, e.g., [37]). Here we work with the following.

▶ **Definition 8** (Congestion[20])**.** Let $T_{\{1,3\}}$ be the set of unrooted binary trees.[21] The *congestion* $\mathrm{cng}(G)$ of a graph G is defined as

$$\mathrm{cng}(G) = \min\{\mathrm{cng}_{H,\mathcal{E}}(G) : H \in T_{\{1,3\}},\ \mathcal{E} = (\iota, \rho) \text{ with } \iota\colon V(G) \to L(H) \text{ bijection}\},$$

where $L(H)$ denotes the set of leaves of H.

In other words, we minimize $\mathrm{cng}_{H,\mathcal{E}}(G)$ when the host graph H is an unrooted binary tree and the mapping ι maps the nodes of G bijectively onto the leaves of H. The routing ρ is uniquely determined as the host graph is a tree.

▶ **Theorem 9** (Bienstock, p. 108–111 of [3])**.** *Given a graph G, we have*[22]

$$\max\left\{\tfrac{2}{3}(\mathrm{tw}(G)+1), \Delta(G)\right\} \leq \mathrm{cng}(G) \leq \Delta(G)(\mathrm{tw}(G)+1).$$

---

[19] The inequality $\mathrm{cw}(G) \leq \Delta(G)\,\mathrm{pw}(G)$ seems to be already present in the earlier work of Chung and Seymour [15] on the relation of cutwidth to another parameter called topological bandwith (see Theorem 2 in [15]). Pathwidth plays an intermediate, connecting role there. However, the inequality is phrased and proved explicitly by Bodlaender in [6].

[20] It is important to note that congestion in the sense of Definition 8 is also known as *carving width*, a term which was coined by Robertson and Seymour in [46]. However, the usual abbreviation for carving width is 'cw' which clashes with that of the cutwidth. Therefore we stick to the name 'congestion' and the abbreviation 'cng' to avoid this potential confusion in notation.

[21] An *unrooted binary tree* is a tree in which each node is incident to either one or three arcs.

[22] Only the right-hand side inequality of Theorem 9, $\mathrm{cng}(G) \leq (\mathrm{tw}(G)+1)\Delta(G)$, is formulated explicitly in [3] as Theorem 1 on p. 111, whereas the left-hand side inequality is stated "inline" in the preceding paragraphs on the same page.

See the full version of the article for a comparison of cutwidth, pathwidth, treewidth and congestion of the Petersen graph. Also, see the full version for an explanation of how width-type parameters are used in FPT-algorithms in computational topology, and a comparison of different parameters for their potential computational advantages or disadvantages.

4 Thin position, and non-Haken 3-manifolds of large genus

In [42] Scharlemann and Thompson extend the concept of thin-position from knot theory [20] to 3-manifolds and define the linear width of a manifold.[23] For this they look at decompositions of a 3-manifold $\mathcal{M}$ into pairs of compression bodies, separated by so-called *large* boundary surfaces. This setup is explained in detail at the end of Section 2.2.

Given such a decomposition of a 3-manifold $\mathcal{M}$ into s pairs of compression bodies with large bounding surfaces $\mathcal{R}_i$, $1 \leq i \leq s$, consider the multiset $\{c(\mathcal{R}_i) : 1 \leq i \leq s\}$, where $c(\mathcal{S}) = \max\{0, 2g(\mathcal{S}) - 1\}$ for a connected surface $\mathcal{S}$,[24] and $c(\mathcal{S}) = \sum_j c(S_j)$ for a surface $\mathcal{S}$ with connected components S_j. This multiset $\{c(\mathcal{R}_i) : 1 \leq i \leq s\}$, when arranged in a decreasing order, is called the *linear width* of the decomposition. The *linear width* of a manifold $\mathcal{M}$, denoted by $\mathcal{L}(\mathcal{M})$, is defined to be the lexicographically smallest width of all such decompositions of $\mathcal{M}$. A manifold $\mathcal{M}$ together with a decomposition into compression bodies realizing $\mathcal{L}(\mathcal{M})$ is said to be in *thin position*.

A guiding idea behind thin position is to attach 2- and 3-handles as early as possible and 0- and 1-handles as late as possible in order to obtain a decomposition for which the "topological complexity" of the large bounding surfaces is minimized.

▶ **Theorem 10** (Scharlemann–Thompson [42]). *Let $\mathcal{M}$ be a 3-manifold together with a decomposition into compression bodies $(\mathcal{N}_1, \mathcal{K}_1, \mathcal{N}_2, \mathcal{K}_2, \ldots, \mathcal{N}_s, \mathcal{K}_s)$ in thin position, and let $\mathcal{F}_i \subset \mathcal{M}$, $1 \leq i < s$, be the set of small bounding surfaces as defined in Section 2.2. Then every connected component of every surface $\mathcal{F}_i$ is incompressible.*

Theorem 10 has the following consequence (see the full version of the article for a sketch of the proof).

▶ **Theorem 11** (Scharlemann–Thompson [42]). *Let $\mathcal{M}$ be irreducible, non-Haken. Then the smallest width decomposition of $\mathcal{M}$ into compression bodies is a Heegaard decomposition of minimal genus $\mathfrak{g}(\mathcal{M})$. In particular, the linear width of $\mathcal{M}$ is given by a list containing only one element, namely $\mathcal{L}(\mathcal{M}) = (2\mathfrak{g}(\mathcal{M}) - 1)$.*

The next theorem of Agol provides an infinite family of 3-manifolds for which we can apply our results established in the subsequent sections.

▶ **Theorem 12** (Agol, Theorem 3.2 in [1]). *There exist orientable, closed, irreducible, and non-Haken 3-manifolds of arbitrarily large Heegaard genus.*

▶ Remark. The construction used to prove Theorem 12 starts with non-Haken n-component link complements, and performs Dehn fillings which neither create incompressible surfaces, nor decrease the (unbounded) Heegaard genera of the complements. The existence of such Dehn fillings is guaranteed by work due to Hatcher [23] and Moriah–Rubinstein [36]. As can be deduced from the construction, the manifolds in question are closed and orientable.

[23] Also see [25] and the textbook [41] for an introduction to generalized Heegaard splittings and to thin position, and for a survey of recent results.

[24] $g(\mathcal{S}) = 1 - \chi(\mathcal{S})/2$ denotes the (orientable) genus, and $\chi(\mathcal{S})$ is the Euler characteristic of $\mathcal{S}$.

5 An obstruction to bounded cutwidth and pathwidth

In this section we establish an upper bound for the Heegaard genus of a 3-manifold $\mathcal{M}$ in terms of the pathwidth of any triangulation of $\mathcal{M}$ (cf. Theorem 3). As an application of this bound we prove Theorem 1. That is, we show that there exists an infinite family of 3-manifolds not admitting triangulations of uniformly bounded pathwidth.

▶ **Theorem 13.** *Let $\mathcal{M}$ be a 3-manifold of linear width $\mathcal{L}(\mathcal{M})$ with dominant entry $\mathcal{L}(\mathcal{M})_1$. Furthermore, let $\mathcal{T}$ be a triangulation of $\mathcal{M}$ with dual graph $\Gamma(\mathcal{T})$ of cutwidth $\mathrm{cw}(\Gamma(\mathcal{T}))$. Then $\mathcal{L}(\mathcal{M})_1 \leq 6\,\mathrm{cw}(\Gamma(\mathcal{T})) + 7$.*

Proof of Theorem 3 assuming Theorem 13. By Theorem 7, $\mathrm{cw}(\Gamma(\mathcal{T})) \leq 4\,\mathrm{pw}(\Gamma(\mathcal{T}))$ since dual graphs of 3-manifold triangulations are 4-regular. By Theorem 11, $\mathcal{L}(\mathcal{M})_1 = 2\mathfrak{g}(\mathcal{M}) - 1$ whenever $\mathcal{M}$ is irreducible and non-Haken. Combining these relations with the inequality provided by Theorem 13 yields the result. ◀

Theorem 1 is now obtained from Theorem 3 and Agol's Theorem 12. It remains to prove Theorem 13. We begin with a basic, yet very useful definition.

▶ **Definition 14.** Let $\mathcal{T}$ be a triangulations of a 3-manifold $\mathcal{M}$. The *canonical handle decomposition* $\mathrm{chd}(\mathcal{T})$ of $\mathcal{T}$ is given by

- one 0-handle for the interior of each tetrahedron of $\mathcal{T}$,
- one 1-handle for a thickened version of the interior of each triangle of $\mathcal{T}$,
- one 2-handle for a thickened version of the interior of each edge of $\mathcal{T}$, and
- one 3-handle for a neighborhood of each vertex of $\mathcal{T}$.

The following lemma gives a bound on the complexity of boundary surfaces occurring in the process of building up a manifold $\mathcal{M}$ from the handles of the canonical handle decomposition of a given triangulation of $\mathcal{M}$ (see the full version of the article for a proof).

▶ **Lemma 15.** *Let $\mathcal{T}$ be a (semi-simplicial) triangulation of a 3-manifold $\mathcal{M}$ and let $\Delta_1 < \Delta_2 < \ldots < \Delta_n \in \mathcal{T}$ be a linear ordering of its tetrahedra. Moreover, let $\mathcal{H}_1 \subset \mathcal{H}_2 \subset \ldots \subset \mathcal{H}_n = \mathrm{chd}(\mathcal{T})$ be a filtration of $\mathrm{chd}(\mathcal{T})$ where $\mathcal{H}_j \subset \mathrm{chd}(\mathcal{T})$ is the codimension zero submanifold consisting of all handles of $\mathrm{chd}(\mathcal{T})$ disjoint from tetrahedra Δ_i, $i > j$. Then passing from $\mathcal{H}_j$ to $\mathcal{H}_{j+1}$ corresponds to adding at most 15 handles, with the maximum of the sum of the genera of the components of any of the bounding surfaces occurring in the process being no larger than the sum of the genera of the components of $\partial\mathcal{H}_j$ plus four.*

With the help of Lemma 15 we can now prove Theorem 13.

Proof of Theorem 13. Let v_j, $1 \leq j \leq n$, be the nodes of $\Gamma(\mathcal{T})$ with corresponding tetrahedra $\Delta_j \in \mathcal{T}$, $1 \leq j \leq n$. We may assume, without loss of generality, that the largest cutset of the linear ordering $v_1 < v_2 < \ldots < v_n$ has cardinality $\mathrm{cw}(\Gamma(\mathcal{T})) = k$.

Let $\mathcal{H}_j \subset \mathrm{chd}(\mathcal{T})$, $1 \leq j \leq n$, be the filtration from Lemma 15. Moreover, let C_j, $1 \leq j < n$, be the cutsets of the linear ordering above. Naturally, the cutset C_j can be associated with at most k triangles of $\mathcal{T}$ with, together, at most $3k$ edges and at most $3k$ vertices of $\mathcal{T}$. Let $\mathcal{H}(C_j) \subset \mathrm{chd}(\mathcal{T})$ be the corresponding submanifold formed from the at most k 1-handles and at most $3k$ 2- and 3-handles each of $\mathrm{chd}(\mathcal{T})$ associated with these faces of $\mathcal{T}$.

By construction, the boundary $\partial\mathcal{H}(C_j)$ of $\mathcal{H}(C_j)$ decomposes into two parts, one of which coincides with the boundary surface $\partial\mathcal{H}_j$. Since $\mathcal{H}(C_j)$ is of the form "neighborhood of k triangles in $\mathcal{T}$", and since the 2- and 3-handles of $\mathrm{chd}(\mathcal{T})$ form a handlebody, the 2- and 3-handles of $\mathcal{H}(C_j)$ form a collection of handlebodies with sum of genera at most $3k$.

To complete the construction of $\mathcal{H}(C_j)$, the remaining at most k 1-handles are attached to this collection of handlebodies as 2-handles. These either increase the number of boundary surface components, or decrease the overall sum of genera of the boundary components. Altogether, the sum of genera of $\partial\mathcal{H}_j \subset \partial\mathcal{H}(C_j)$ is bounded above by $3k$.

Hence, following Lemma 15, the sum g of genera of the components of any bounding surface for any sequence of handle attachments of $\mathrm{chd}(\mathcal{T})$ compatible with the ordering $v_1 < v_2 < \ldots < v_n$ is bounded above by $3\,\mathrm{cw}(\Gamma(\mathcal{T}))+4$. It follows that $2g-1 \leq 6\,\mathrm{cw}(\Gamma(\mathcal{T}))+7$, and finally, by the definition, $\mathcal{L}(\mathcal{M})_1 \leq 6\,\mathrm{cw}(\Gamma(\mathcal{T})) + 7$. ◀

6 An obstruction to bounded congestion and treewidth

The goal of this section is to prove Theorems 2 and 4, the counterparts of Theorem 1 and 3 for treewidth. At the core of the proof is the following explicit connection between the congestion of the dual graph of any triangulation of a 3-manifold $\mathcal{M}$ and its Heegaard genus.

▶ **Theorem 16.** *Let $\mathcal{M}$ be an orientable, closed, irreducible, non-Haken 3-manifold which has a triangulation $\mathcal{T}$ with dual graph $\Gamma(\mathcal{T})$ of congestion $\mathrm{cng}(\Gamma(\mathcal{T})) \leq k$. Then either $\mathcal{M}$ has Heegaard genus $\mathfrak{g}(\mathcal{M}) < 12k$, or $\mathcal{T}$ only contains one tetrahedron.*

Proof of Theorem 4 assuming Theorem 16. Since dual graphs of 3-manifold triangulations are 4-regular, Theorem 9 implies $\mathrm{cng}(\Gamma(\mathcal{T})) \leq 4(\mathrm{tw}(\Gamma(\mathcal{T})) + 1)$. Moreover, the only closed orientable 3-manifolds which can be triangulated with a single tetrahedron are the 3-sphere of Heegaard genus zero, and the lens spaces of type $\mathrm{L}(4,1)$ and $\mathrm{L}(5,2)$ of Heegaard genus one. ◀

Similarly as before, Theorem 2 immediately follows by combining Theorems 4 and 12. Hence, the remainder of the section is dedicated to the proof of Theorem 16.

Proof of Theorem 16. Let $\mathcal{M}$ be an orientable, closed, irreducible, non-Haken 3-manifold which has a triangulation $\mathcal{T}$ whose dual graph $\Gamma(\mathcal{T})$ has congestion $\mathrm{cng}(\Gamma(\mathcal{T})) \leq k$, and let T be an unrooted binary tree realizing $\mathrm{cng}(\Gamma(\mathcal{T})) \leq k$. If $k = 0$, $\mathcal{T}$ must consist of a single tetrahedron, and the theorem holds. Thus we can assume that $k \geq 1$. Moreover, let $\mathrm{chd}(\mathcal{T})$ be the canonical handle decomposition of $\mathcal{T}$ as defined in Definition 14.

For every $e \in E(T)$, there exist arcs $\gamma_1, \gamma_2, \ldots, \gamma_\ell \in E(\Gamma(\mathcal{T}))$, $\ell \leq k$, running parallel to e, corresponding to triangles $t_1, t_2, \ldots, t_\ell$ in $\mathcal{T}$. Let $\mathcal{H}_e \subset \mathrm{chd}(\mathcal{T})$ be the submanifold built from the ℓ 1-handles of $\mathrm{chd}(\mathcal{T})$ corresponding to the triangles t_i, $1 \leq i \leq \ell$, and the $\leq 3\ell$ 2- and 3-handles each of $\mathrm{chd}(\mathcal{T})$ corresponding to the edges and vertices of these triangles.

▶ **Lemma 17.** *The codimension zero submanifold $\mathrm{chd}(\mathcal{T}) \setminus \mathcal{H}_e$ decomposes into a pair of disjoint components, denoted by $\mathcal{H}_e^+$ and $\mathcal{H}_e^-$.*

See Figure 1(i), and, for the proof, the full version of the article.

▶ **Lemma 18.** *At least one of $\mathcal{H}_e^+$ or $\mathrm{chd}(\mathcal{T}) \setminus \mathcal{H}_e^+$ can be built from at most $5k$ handles. The same statement holds for $\mathcal{H}_e^-$ and $\mathrm{chd}(\mathcal{T}) \setminus \mathcal{H}_e^-$.*

Idea of the proof of Lemma 18. Consider the decomposition $\mathcal{M} = \mathcal{H}_e^+ \cup_{\partial\mathcal{H}_e^+} (\mathrm{chd}(\mathcal{T}) \setminus \mathcal{H}_e^+) =: \mathcal{A}_0 \cup_{\mathcal{S}_0} \mathcal{B}_0$. Both the number of connected components and the genus of the surface $\mathcal{S}_0 = \partial\mathcal{H}_e^+$ is in $O(k)$, as $\mathcal{H}_e$ can be built from $O(k)$ handles (see above), and $\mathcal{S}_0 = \partial\mathcal{H}_e^+ \subset \partial\mathcal{H}_e$. Since $\mathcal{M}$ is irreducible, non-Haken, and all surface components of $\mathcal{S}_0$ are 2-sided by construction, none of them can be incompressible. Therefore, via a sequence of handle reductions along carefully chosen compressing disks we get a decomposition

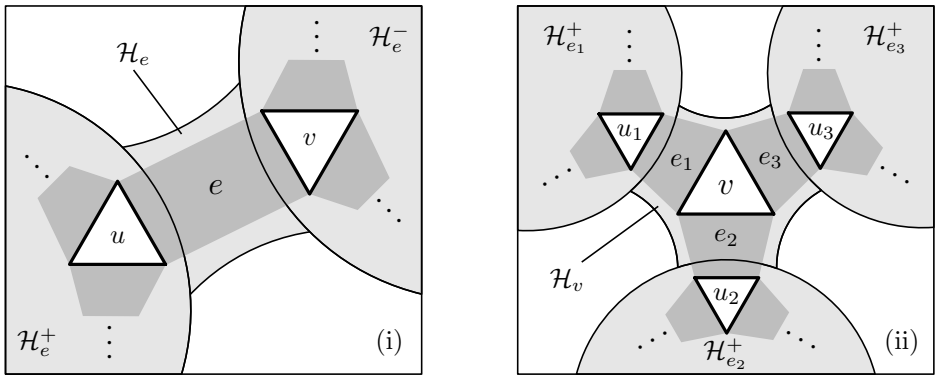


Figure 1 Schematic overview of the decompositions of $\mathcal{M}$ into (i) three and (ii) four submanifolds at arcs and degree-three nodes of the tree T, respectively

$\mathcal{M} = \mathcal{A}_* \cup_{\mathcal{S}_*} \mathcal{B}_*$ in $O(k)$ steps, where $\mathcal{S}_*$ consists of $m \in O(k)$ copies of the 2-sphere. It turns out that either $\mathcal{A}_*$ or $\mathcal{B}_*$, say $\mathcal{B}_*$, is a collection of punctured 3-balls with an overall number of punctures being equal to m, which implies that $\mathcal{B}_*$ can be built from m handles, from which $\mathcal{B}_0$ can be recovered via $O(k)$ handle attachments. We can make these bounds explicit by careful bookkeeping to obtain the result. See the full version of the article for a more detailed proof. ◀

Now at every degree three node $v \in V(T)$ we define a decomposition of $\mathcal{M}$ into four submanifolds (Figure 1(ii)). Three of them, $\mathcal{H}_{e_1}^+$, $\mathcal{H}_{e_2}^+$, and $\mathcal{H}_{e_3}^+$, are arising from the three connected components of T after removing the three arcs incident to v (cf. Lemma 17), and the fourth submanifold, $\mathcal{H}_v$, contains all remaining handles of $\mathrm{chd}(\mathcal{T})$.

By Lemma 18, at least two of the $\mathcal{H}_{e_i}^+$, $1 \leq i \leq 3$, can be built from at most $5k$ handles. For the fourth submanifold we have $\mathcal{H}_v = \mathcal{H}_{e_1} \cup \mathcal{H}_{e_2} \cup \mathcal{H}_{e_3}$. Observe that at most k arcs of $\Gamma(\mathcal{T})$ run parallel to each of the $e_i \in E(T)$. Counting those arcs along the e_i we encounter each of them twice, therefore at most $\frac{3}{2}k$ arcs of $\Gamma(\mathcal{T})$ meet v. It follows that $\mathcal{H}_v$ is a collection of at most $\frac{3}{2}k$ handlebodies with sum of genera at most $\frac{9}{2}k$ and an additional at most $\frac{3}{2}k$ 2-handles attached (see full version for details). Altogether, $\mathcal{H}_v$ can be built from at most $\frac{3}{2}k + \frac{9}{2}k + \frac{3}{2}k < 8k$ handles. Moreover, if v has a neighbor which is a leaf of T, then its corresponding submanifold is obtained from a 1-tetrahedron submanifold of $\mathcal{T}$ with at most three interior faces (the tetrahedron itself, at most one triangle, and at most one edge). Hence, this part corresponds to at most three handles of $\mathrm{chd}(\mathcal{T})$.

If all $\mathcal{H}_{e_i}^+$, $1 \leq i \leq 3$, are of size at most $5k$, $\mathcal{M}$ can be built using less than $(3{\cdot}5{+}8)k = 23k$ handles, thus $\mathfrak{g}(\mathcal{M}) < 12k$ and we are done. Hence, assume that exactly one of them, say $\mathcal{H}_{e_1}^+$, requires a larger number of handles to be built. Let u_1 be the other node of e_1.

Now we choose u_1 to be the new center (instead of v) and repeat the above process. Moving from v to u_1 merges $\mathcal{H}_{e_2}^+$, $\mathcal{H}_{e_3}^+$, and $\mathcal{H}_v$ (note that each of $\mathcal{H}_{e_2}^+$ and $\mathcal{H}_{e_3}^+$ can be built with $\leq 5k$ handles, and $\mathcal{H}_v$ can be built with $< 8k$ handles), and splits the remaining larger part into three submanifolds. Since these three submanifolds together form the larger part, they either cannot all be built from at most $5k$ / less than $8k$ handles – or $\mathcal{M}$ can be built from at most $23k$ handles. Similarly, the three merged parts must be one of the two new submanifolds which can be built from at most $5k$ handles – or $\mathcal{M}$ can be built from at most $23k$ handles. In both cases it follows that $\mathfrak{g}(\mathcal{M}) < 12k$ and we are done. Hence, assume one of the submanifolds coming from one of the two new subtrees must require a larger number of handles to be built. However, at the same time, the connected component of

$T \setminus \mathrm{st}_T(u_1)$ corresponding to this submanifold of unbounded size has fewer nodes than that corresponding to the previously largest part. Iterating the process must thus eventually lead to the conclusion that $\mathcal{M}$ is of Heegaard genus less than $12k$. ◀

References

1. I. Agol. Small 3-manifolds of large genus. *Geom. Dedicata*, 102:53–64, 2003. doi:10.1023/B:GEOM.0000006584.85248.c5.
2. L. Bessières, G. Besson, S. Maillot, M. Boileau, and J. Porti. *Geometrisation of 3-manifolds*, volume 13 of *EMS Tracts in Mathematics*. European Mathematical Society (EMS), Zürich, 2010. doi:10.4171/082.
3. D. Bienstock. On embedding graphs in trees. *J. Comb. Theory, Ser. B*, 49(1):103–136, 1990. doi:10.1016/0095-8956(90)90066-9.
4. D. Bienstock. Graph searching, path-width, tree-width and related problems (A survey). In *Reliability Of Computer And Communication Networks*, volume 5 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 33–50. Amer. Math. Soc., Providence, RI, 1991. doi:10.1090/dimacs/005/02.
5. H. L. Bodlaender. A tourist guide through treewidth. *Acta Cybern.*, 11(1-2):1–21, 1993. URL: http://www.inf.u-szeged.hu/actacybernetica/edb/vol11n1_2/starten.xml.
6. H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theor. Comput. Sci.*, 209(1-2):1–45, 1998. doi:10.1016/S0304-3975(97)00228-4.
7. H. L. Bodlaender. Discovering treewidth. In *SOFSEM 2005: Theory and Practice of Computer Science, 31st Conference on Current Trends in Theory and Practice of Computer Science, Liptovský Ján, Slovakia, January 22–28, 2005, Proceedings*, pages 1–16, 2005. doi:10.1007/978-3-540-30577-4_1.
8. H. L. Bodlaender and A. M. C. A. Koster. Combinatorial optimization on graphs of bounded treewidth. *Comput. J.*, 51(3):255–269, 2008. doi:10.1093/comjnl/bxm037.
9. B. A. Burton, R. Budney, W. Pettersson, et al. Regina: Software for 3-manifold topology and normal surface theory, 1999–2017. URL: http://regina-normal.github.io.
10. B. A. Burton and R. G. Downey. Courcelle's theorem for triangulations. *J. Comb. Theory, Ser. A*, 146:264–294, 2017. doi:10.1016/j.jcta.2016.10.001.
11. B. A. Burton, H. Edelsbrunner, J. Erickson, and S. Tillmann, editors. *Computational Geometric and Algebraic Topology*, volume 12 of *Oberwolfach Reports*. EMS Publishing House, 2015. doi:10.4171/OWR/2015/45.
12. B. A. Burton, T. Lewiner, J. Paixão, and J. Spreer. Parameterized complexity of discrete Morse theory. *ACM Trans. Math. Softw.*, 42(1):6:1–6:24, 2016. doi:10.1145/2738034.
13. B. A. Burton, C. Maria, and J. Spreer. Algorithms and complexity for Turaev–Viro invariants. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6–10, 2015, Proceedings, Part I*, pages 281–293, 2015. doi:10.1007/978-3-662-47672-7_23.
14. B. A. Burton and J. Spreer. The complexity of detecting taut angle structures on triangulations. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6–8, 2013*, pages 168–183, 2013. doi:10.1137/1.9781611973105.13.
15. F. R. K. Chung and P. D. Seymour. Graphs with small bandwidth and cutwidth. *Discrete Mathematics*, 75(1–3):113–119, 1989. doi:10.1016/0012-365X(89)90083-6.
16. B. Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990. doi:10.1016/0890-5401(90)90043-H.
17. R. Diestel. *Graph theory*, volume 173 of *Graduate Texts in Mathematics*. Springer, Berlin, fifth edition, 2017. doi:10.1007/978-3-662-53622-3.

18 R. G. Downey and M. R. Fellows. *Parameterized complexity*. Monographs in Computer Science. Springer-Verlag, New York, 1999. doi:10.1007/978-1-4612-0515-9.

19 R. G. Downey and M. R. Fellows. *Fundamentals of parameterized complexity*. Texts in Computer Science. Springer, London, 2013. doi:10.1007/978-1-4471-5559-1.

20 D. Gabai. Foliations and the topology of 3-manifolds. III. *J. Differential Geom.*, 26(3):479–536, 1987. doi:10.4310/jdg/1214441488.

21 W. Haken. Theorie der Normalflächen. *Acta Math.*, 105:245–375, 1961. doi:10.1007/BF02559591.

22 J. Hass, J. C. Lagarias, and N. Pippenger. The computational complexity of knot and link problems. *J. ACM*, 46(2):185–211, 1999. doi:10.1145/301970.301971.

23 A. E. Hatcher. On the boundary curves of incompressible surfaces. *Pacific J. Math.*, 99(2):373–377, 1982. doi:10.2140/pjm.1982.99.373.

24 P. Hlinený, S. Oum, D. Seese, and G. Gottlob. Width parameters beyond tree-width and their applications. *Comput. J.*, 51(3):326–362, 2008. doi:10.1093/comjnl/bxm052.

25 H. Howards, Y. Rieck, and J. Schultens. Thin position for knots and 3-manifolds: a unified approach. In *Workshop on Heegaard Splittings*, volume 12 of *Geom. Topol. Monogr.*, pages 89–120. Geom. Topol. Publ., Coventry, 2007. doi:10.2140/gtm.2007.12.89.

26 K. Huszár, J. Spreer, and U. Wagner. On the treewidth of triangulated 3-manifolds, 2017. arXiv:1712.00434.

27 S. V. Ivanov. The computational complexity of basic decision problems in 3-dimensional topology. *Geom. Dedicata*, 131:1–26, 2008. doi:10.1007/s10711-007-9210-4.

28 R. Kirby and P. Melvin. Local surgery formulas for quantum invariants and the Arf invariant. In *Proceedings of the Casson Fest*, volume 7 of *Geom. Topol. Monogr.*, pages 213–233. Geom. Topol. Publ., Coventry, 2004. doi:10.2140/gtm.2004.7.213.

29 B. Kleiner and J. Lott. Notes on Perelman's papers. *Geom. Topol.*, 12(5):2587–2855, 2008. doi:10.2140/gt.2008.12.2587.

30 G. Kuperberg. Knottedness is in NP, modulo GRH. *Adv. Math.*, 256:493–506, 2014. doi:10.1016/j.aim.2014.01.007.

31 G. Kuperberg. Algorithmic homeomorphism of 3-manifolds as a corollary of geometrization, 2015. arXiv:1508.06720.

32 M. Lackenby. The efficient certification of knottedness and Thurston norm, 2016. arXiv:1604.00290.

33 M. Lackenby. Some conditionally hard problems on links and 3-manifolds. *Discrete & Computational Geometry*, 58(3):580–595, 2017. doi:10.1007/s00454-017-9905-8.

34 C. Maria and J. Spreer. A polynomial time algorithm to compute quantum invariants of 3-manifolds with bounded first Betti number. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16–19*, pages 2721–2732, 2017. doi:10.1137/1.9781611974782.180.

35 E. E. Moise. Affine structures in 3-manifolds. V. The triangulation theorem and Hauptvermutung. *Ann. of Math. (2)*, 56:96–114, 1952. doi:10.2307/1969769.

36 Y. Moriah and H. Rubinstein. Heegaard structures of negatively curved 3-manifolds. *Comm. Anal. Geom.*, 5(3):375–412, 1997. doi:10.4310/CAG.1997.v5.n3.a1.

37 M. I. Ostrovskii. Minimal congestion trees. *Discrete Mathematics*, 285(1–3):219–226, 2004. doi:10.1016/j.disc.2004.02.009.

38 N. Robertson and P. D. Seymour. Graph minors. I. Excluding a forest. *J. Comb. Theory, Ser. B*, 35(1):39–61, 1983. doi:10.1016/0095-8956(83)90079-5.

39 N. Robertson and P. D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *J. Algorithms*, 7(3):309–322, 1986. doi:10.1016/0196-6774(86)90023-4.

40 J. H. Rubinstein. An algorithm to recognize the 3-sphere. In S. D. Chatterji, editor, *Proceedings of the International Congress of Mathematicians: August 3–11, 1994, Zürich, Switzerland*, volume 1, pages 601–611. Birkhäuser Verlag, 1995.

41 M. Scharlemann, J. Schultens, and T. Saito. *Lecture notes on generalized Heegaard splittings*. World Scientific Publishing Co. Pte. Ltd., Hackensack, NJ, 2016. Three lectures on low-dimensional topology in Kyoto. doi:10.1142/10019.

42 M. Scharlemann and A. Thompson. Thin position for 3-manifolds. In *Geometric topology (Haifa, 1992)*, volume 164 of *Contemp. Math.*, pages 231–238. Amer. Math. Soc., Providence, RI, 1994. doi:10.1090/conm/164/01596.

43 S. Schleimer. Sphere recognition lies in NP. In *Low-dimensional and symplectic topology*, volume 82 of *Proc. Sympos. Pure Math.*, pages 183–213. Amer. Math. Soc., Providence, RI, 2011. doi:10.1090/pspum/082/2768660.

44 J. Schultens. *Introduction to 3-manifolds*, volume 151 of *Graduate Studies in Mathematics*. Amer. Math. Soc., Providence, RI, 2014. doi:10.1090/gsm/151.

45 P. Scott and H. Short. The homeomorphism problem for closed 3-manifolds. *Algebr. Geom. Topol.*, 14(4):2431–2444, 2014. doi:10.2140/agt.2014.14.2431.

46 P. D. Seymour and R. Thomas. Call routing and the ratcatcher. *Combinatorica*, 14(2):217–241, 1994. doi:10.1007/BF01215352.

47 A. Thompson. Thin position and the recognition problem for S^3. *Math. Res. Lett.*, 1(5):613–630, 1994. doi:10.4310/MRL.1994.v1.n5.a9.

48 W. P. Thurston. *Three-dimensional geometry and topology. Vol. 1*, volume 35 of *Princeton Mathematical Series*. Princeton University Press, Princeton, NJ, 1997. Edited by Silvio Levy. doi:10.1515/9781400865321.

49 R. Zentner. Integer homology 3-spheres admit irreducible representations in $SL(2,\mathbb{C})$, 2016. arXiv:1605.08530.

On Partial Covering For Geometric Set Systems

Tanmay Inamdar
Department of Computer Science, University of Iowa,
Iowa City, IA, USA.
tanmay-inamdar@uiowa.edu

Kasturi Varadarajan
Department of Computer Science, University of Iowa,
Iowa City, IA, USA.
kasturi-varadarajan@uiowa.edu

Abstract

We study a generalization of the Set Cover problem called the *Partial Set Cover* in the context of geometric set systems. The input to this problem is a set system $(X, \mathcal{R})$, where X is a set of elements and $\mathcal{R}$ is a collection of subsets of X, and an integer $k \leq |X|$. Each set in $\mathcal{R}$ has a non-negative weight associated with it. The goal is to cover at least k elements of X by using a minimum-weight collection of sets from $\mathcal{R}$. The main result of this article is an LP rounding scheme which shows that the integrality gap of the Partial Set Cover LP is at most a constant times that of the Set Cover LP for a certain projection of the set system $(X, \mathcal{R})$. As a corollary of this result, we get improved approximation guarantees for the Partial Set Cover problem for a large class of geometric set systems.

2012 ACM Subject Classification Theory of computation → Packing and covering problems, Theory of computation → Rounding techniques, Theory of computation → Computational geometry, Mathematics of computing → Approximation algorithms

Keywords and phrases Partial Set Cover, Geometric Set Cover

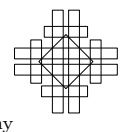
Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.47

Funding This material is based upon work supported by the National Science Foundation under Grants CCF-1318996 and CCF-1615845.

1 Introduction

In the Set Cover (SC) problem, the input is a set system $(X, \mathcal{R})$, where X is a set of n elements, and $\mathcal{R}$ is a collection of subsets of X . The goal is to find a minimum-size collection $\mathcal{R}' \subseteq \mathcal{R}$ that *covers* X, i.e., the union of the sets in $\mathcal{R}'$ contains the elements of X. In the weighted version, each set $S_i \in \mathcal{R}$ has a non-negative weight w_i associated with it, and we seek to minimize the weight of $\mathcal{R}'$. A simple greedy algorithm finds a solution that is guaranteed to be within $O(\log n)$ factor from the optimal (see [33] for references), and it is not possible to do better in general using any polynomial-time algorithm , under certain standard complexity theoretic assumptions [15, 10]. In the rest of this article, we assume polynomial running time in any statement that we make about an algorithm.

The question of whether we can improve the $O(\log n)$ bound has been extensively studied for geometric set systems. We focus on three important classes – covering, hitting, and art gallery problems. In the Geometric Set Cover problem, X typically consists of points in $\mathbb{R}^d$, and $\mathcal{R}$ contains sets induced by a certain class of geometric objects via containment. For example, each set in $\mathcal{R}$ might be the subset of X contained in a hypercube. Some of the well-studied examples include covering points by disks in the plane, fat triangles, etc. In the




Geometric Hitting Set problem, X is a set of geometric objects, and each set in $\mathcal{R}$ is the subset consisting of all objects in X that are stabbed by some point. In an example of the art gallery problem, X consists of a set of points in a simple polygon, and each set in $\mathcal{R}$ is the subset consisting of all points in X that can be seen by some vertex of the polygon [25]. Thus, the set system here is defined by visibility.

For many such geometric set systems, it is possible to obtain approximation guarantees better than $O(\log n)$. We survey two of the main approaches to obtain such guarantees. The first and the most successful approach is based on the SC Linear Program (LP) and its connection to ε-nets. For completeness, we state the standard SC LP for the weighted case.

$$\text{minimize} \sum_{S_i \in \mathcal{R}} w_i x_i$$

$$\text{subject to} \sum_{i: e_j \in S_i} x_i \geq 1, \qquad e_j \in X \tag{1}$$

$$x_i \geq 0, \qquad S_i \in \mathcal{R} \tag{2}$$

For the unweighted case, Even et al. [13] showed that, if for a certain set system, $O\left(\frac{1}{\varepsilon} \cdot g\left(\frac{1}{\varepsilon}\right)\right)$ size ε-nets exist, then the integrality gap of the SC LP is $O(g(OPT))$, where OPT is the size of the optimal solution. This result is constructive, in that an efficient algorithm for constructing ε-nets also yields an efficient algorithm for obtaining an $O(g(OPT))$ approximation. (A similar result was obtained earlier by Brönnimann and Goodrich [3], without using LP machinery). It is fairly well-known ([8, 20]) that, for a large class of geometric set systems, ε-nets of size $O\left(\frac{1}{\varepsilon} \log\left(\frac{1}{\varepsilon}\right)\right)$ can be computed efficiently, which implies $O(\log(OPT))$ approximation for the set cover problem on the corresponding geometric set system. Clarkson and Varadarajan [9] showed that if the *union complexity* of any set of n objects is $O(n \cdot h(n))$, then ε-nets of size $O\left(\frac{1}{\varepsilon} \cdot h\left(\frac{1}{\varepsilon}\right)\right)$ exist. Aronov et al. [1] gave a tighter bound of $O(\frac{1}{\varepsilon} \cdot \log h(\frac{1}{\varepsilon}))$ on the size of ε-nets for the objects of union complexity $O(n \cdot h(n))$ (see also [31]). Some of these results were extended to the weighted case in [32, 6] by a technique called *quasi-uniform sampling*. We summarize some of these ε-net based results for the set cover problem for geometric set systems in Table 1.

Another approach for tackling SC for geometric set systems is by combinatorial algorithms. The dominant paradigm from this class is a simple local search algorithm. The effectiveness of local search was first demonstrated by Mustafa and Ray [29], who gave the first PTAS for covering points by disks in plane. There have been a series of results that build on their work, culminating in Govindarajan et al. [19], who show that local search yields a PTAS for SC for a fairly general class of objects such as pseudodisks and non-piercing regions in plane. Krohn et al. [27] gave a PTAS for the terrain guarding problem, where the geometric set

Table 1 LP-based approximation ratios for SC. See [9, 1, 32, 14, 6] for the references establishing these bounds. Except for stabbing rectangles in $\mathbb{R}^3$ by points, these bounds hold for the weighted SC. For these problems, we obtain analogous results for weighted PSC.

X	Geometric objects inducing $\mathcal{R}$	Integrality Gap of SC LP
Points in $\mathbb{R}^2$	Disks (via containment)	$O(1)$
	Fat triangles (containment)	$O(\log \log^* n)$
Points in $\mathbb{R}^3$	Unit cubes (containment)	$O(1)$
	Halfspaces (containment)	$O(1)$
Rectangles in $\mathbb{R}^3$	Points (via stabbing)	$O(\log \log n)$
Points on 1.5D terrain	Points on terrain (via visibility)	$O(1)$ [11]

system is defined by visibility. These results for local search only hold for the unweighted set cover problem. Another common strategy, called the *shifting strategy*, was introduced by Hochbaum and Maass [22]. They give a PTAS for covering points by unit balls in $\mathbb{R}^d$; however in this case the set $\mathcal{R}$ consists of *all* unit balls in $\mathbb{R}^d$. Chan [5] gave a PTAS for stabbing a set of fat objects in $\mathbb{R}^d$ using a minimum number of points from $\mathbb{R}^d$. Erlebach and Van Leeuwen [12] combine the shifting strategy with a sophisticated dynamic program to obtain a PTAS for weighted set cover with unit disks in the plane.

Now we turn to the Partial Set Cover (PSC) problem. The input to PSC is the same as that to the SC, along with an additional integer parameter $k \leq |X|$. Here the goal is to cover at least k elements from X while minimizing the size (or weight) of the solution $\mathcal{R}' \subseteq \mathcal{R}$. It is easy to see that PSC is a generalization of SC, and hence it is at least as hard as SC. We note here that another classical problem that is related to both of these problems is the so-called Maximum Coverage (MC) problem. In this problem, we have an upper bound on the number of sets that can be chosen in the solution, and the goal is to cover the maximum number of elements. It is a simple exercise to see that an exact algorithm for the unweighted PSC can be used to solve MC exactly, and vice versa. However the reductions are not approximation-preserving. In particular, the greedy algorithm achieves $1 - 1/e$ approximation guarantee for MC — which is essentially the best possible — whereas it is NP-hard to approximate PSC within $o(\log n)$ factor in general. We refer the reader to [24] for a generalization of MC and a survey of results.

For PSC, the greedy algorithm is shown to be an $O(\log \Delta)$ approximation in [23, 30], where Δ is the size of the largest set in $\mathcal{R}$. Bar-Yehuda [2], using the local ratio technique, and Gandhi et al. [17], using the primal-dual method, give algorithms which achieve an approximation guarantee of f, where f is the maximum frequency of any element in the sets. A special case of PSC is the Partial Vertex Cover (PVC) problem, where we need to pick a minimum size (or weight) subset of vertices that covers at least k edges of the graph. Bshouty and Burroughs [4] and [21] present different approaches for obtaining a 2-approximation based on LP rounding for PVC. We refer the reader to [16] for a more detailed history of these foundational results, and to [28, 26] for more recent results on PVC, PSC, and related problems.

While SC for various geometric set systems has been studied extensively, there is relatively less work studying PSC in the geometric setting. Gandhi et al. [17] give a PTAS for a geometric version of PSC where $\mathcal{R}$ consists of *all* unit disks in the plane. They provide a dynamic program on top of the standard shifting strategy of Hochbaum and Maass [22], thus adapting it for PSC. Using a similar technique, Glaßer et al. [18] give a PTAS for a generalization of partial geometric covering, under a certain assumption on the *density* of the given disks. Chan and Hu [7] give a PTAS for PSC where $\mathcal{R}$ consists of a given set of unit squares in the plane, by combining the shifting strategy with sophisticated dynamic programming.

Our results and techniques

Suppose that we are given a PSC instance $(X, \mathcal{R}, k)$. For any set of elements $Y \subseteq X$, let $\mathcal{R}_{|Y} := \{S \cap Y \mid S \in \mathcal{R}\}$ denote the projected set system. Suppose also that for any projected SC instance $(Y, \mathcal{R}_{|Y})$, (where $Y \subseteq X$) and a corresponding feasible SC LP solution σ_1, we can round σ_1 to a feasible integral SC solution with cost at most β times that of σ_1. That is, we suppose that we can efficiently compute a β-approximation for the SC instance $(Y, \mathcal{R}_{|Y})$ by solving the natural LP relaxation and rounding it. Then, we show that we can round the solution to the natural PSC LP for $(X, \mathcal{R}, k)$ to an integral solution to within a $2\beta + 2$

factor. By the previous discussion about the existence of such rounding algorithms for SC LP for a large class of geometric objects (cf. Table 1), we get the same guarantees for the corresponding PSC instances as well, up to a constant factor. For clarity, we describe a sample of these applications.

1. Suppose we are given a set P of n points and a set $\mathcal{T}$ of *fat* triangles in the plane and a positive weight for each triangle in $\mathcal{T}$. We wish choose a subset $\mathcal{T}' \subseteq \mathcal{T}$ of triangles that covers P, and minimize the weight of $\mathcal{T}'$, defined to be the sum of the weights of the triangles in it. This is a special case of weighted SC obtained by setting $X = P$, and adding the set $t \cap P$ to $\mathcal{R}$ for each triangle in $t \in \mathcal{T}$, with the same weight. There is an $O(\log \log^* n)$ approximation for this problem based on rounding SC LP [14, 6]. We obtain the same approximation guarantee for the partial covering version, where we want a minimum weight subset of $\mathcal{T}$ covering any k of the points in P.

2. Suppose we are given a set $\mathcal{B}$ of n axis-parallel boxes and a set P of points in $\mathbb{R}^3$, and we wish to find a minimum cardinality subset of P that hits (or stabs) each box in $\mathcal{B}$. This a special case of SC obtained by setting $X = \mathcal{B}$, and adding the set $\{b \in \mathcal{B} \mid p \in b\}$ to $\mathcal{R}$ for each point $p \in P$. There is an $O(\log \log n)$ approximation for this problem based on rounding SC LP [1]. Thus, we obtain the same approximation guarantee for the partial version, where we want a minimum cardinality subset of P stabbing any k of the boxes in $\mathcal{B}$.

3. Suppose we have a 1.5D terrain (i.e., an x-monotone polygonal chain in $\mathbb{R}^2$), a set P of points and a set G of n points, called guards, on the terrain along with a positive weight for each guard in G. The goal is to choose a subset $G' \subset G$ such that each point in P is seen by some guard in G, and minimize the weight of G'. Two points p and g on the terrain see each other if the line segment connecting them does not contain a point below the terrain. This is a special case of SC obtained by setting $X = P$, and adding the set $\{p \in P \mid g \text{ sees } p\}$ to $\mathcal{R}$ for each guard $g \in G$. There is an $O(1)$-approximation guarantee for this problem based on rounding SC LP [11]. Thus, we obtain an $O(1)$-approximation for the partial version, where we want a minimum weight subset of G that sees any k of the points in P.

Our algorithm for rounding a solution to the natural PSC LP corresponding to partial cover instance $(X, \mathcal{R}, k)$ proceeds as follows. Let X_1 be the elements that are covered by the LP solution to an extent of at least $1/2$. By scaling the LP solution by a factor of 2, we get a feasible solution to the SC LP corresponding to $(X_1, \mathcal{R}_{|X_1})$, which we round using the LP-based β-approximation algorithm. For the set $X \setminus X_1$, the LP solution provides a total fractional coverage of at least $k - |X_1|$. Crucially, each element of $X \setminus X_1$ is *shallow* in that it is covered to an extent of at most $1/2$. We use this observation to round the LP solution to an integer solution, of at most twice the cost, that covers at least $k - |X_1|$ points of $X \setminus X_1$. This rounding step and its analysis are inspired by the PVC rounding scheme of [4], but there are certain subtleties in adapting it to the PSC problem. To the best of our knowledge, this connection between the SC LP and PSC LP was not observed before.

The rest of this article is organized as follows. In Section 2, we describe the standard LP formulation for the PSC problem, and give an integrality gap example. We describe how to circumvent this integrality gap by preprocessing the input in Section 3. Finally, in Section 4, we describe and analyze the main LP rounding algorithm.

2 Preliminaries

We use the following Integer Programming formulation of PSC (see left side of display below). Here, for each element $e_j \in X$, the variable z_j denotes whether it is one of the k elements that are chosen by the solution. For each such chosen element e_j, the first constraint ensures that at least one set containing it must be chosen. The second constraint ensures that at least k elements are covered by the solution. We relax the integrality Constraints 3, and 4, and formulate it as a Linear Program (see right side).

$$\text{minimize} \sum_{S_i \in \mathcal{R}} w_i x_i$$

$$\text{subject to} \sum_{i:e_j \in S_i} x_i \geq z_j, \quad e_j \in X$$

$$\sum_{e_j \in X} z_j \geq k,$$

$$z_j \in \{0,1\}, \quad e_j \in X \quad (3)$$

$$x_i \in \{0,1\}, \quad S_i \in \mathcal{R} \quad (4)$$

Integer Program

$$\text{minimize} \sum_{S_i \in \mathcal{R}} w_i x_i \quad (5)$$

$$\text{subject to} \sum_{i:e_j \in S_i} x_i \geq z_j, \quad e_j \in X \quad (6)$$

$$\sum_{e_j \in X} z_j \geq k, \quad (7)$$

$$z_j \in [0,1], \quad e_j \in X \quad (8)$$

$$x_i \in [0,1], \quad S_i \in \mathcal{R} \quad (9)$$

Linear Program

Since SC is a special case of PSC where $k = n$, the corresponding LP can be obtained by setting k appropriately in Constraint 7. However, in this case, the LP can be further simplified as described earlier. We denote the cost of a PSC LP solution $\sigma = (x, z)$, for the instance $(X, \mathcal{R})$, as $cost(\sigma) := \sum_{S_i \in \mathcal{R}} w_i x_i$, and the cost of an SC LP solution is defined in exactly the same way. Also, for any collection of sets $\mathcal{R}' \subseteq \mathcal{R}$, we define $w(\mathcal{R}') := \sum_{S_i \in \mathcal{R}'} w_i$. Finally, for a PSC instance $(X, \mathcal{R}, k)$, let $OPT(X, \mathcal{R}, k)$ denote the cost of an optimal solution for that instance.

Unlike SC LP, the integrality gap of PSC LP can be $\Omega(n)$, even for the unweighted case. **Integrality Gap:** Consider the set system $(X, \mathcal{R})$, where $X = \{e_1, \ldots, e_n\}$, and $\mathcal{R} = \{S_1\}$, where $S_1 = X$. Here, $k = 1$, so at least one element has to be covered. The size of the optimal solution is 1, because the only set S_1 has to be chosen. However, consider the following fractional solution $\sigma = (x, z)$, where $z_j = \frac{1}{n}$ for all $e_j \in X$, and $x_1 = \frac{1}{n}$, which has the cost of $\frac{1}{n}$. This shows the integrality gap of n.

However, Gandhi et al. [17] show that after "guessing" the heaviest set in the optimal solution, the integrality gap of the LP corresponding to the residual instance is at most f, where f is the maximum frequency of any element in the set system. In this article, we show that after guessing the heaviest set in the optimal solution, the residual instance has integrality gap at most $2\beta + 2$, where β is the integrality gap of the SC LP for some projection of the same set system.

3 Preprocessing

Henceforth, for convenience, we let $(X', \mathcal{R}', k')$ denote our original input instance of PSC. To circumvent the integrality gap, we preprocess the given instance to "guess" the heaviest set in the optimal solution, and solve the residual instance as in [4, 17] – see Algorithm 1. Let us renumber the sets $\mathcal{R}' = \{S_1, \ldots, S_m\}$, so that $w_1 \leq w_2 \leq \ldots \leq w_m$. For each $S_i \in \mathcal{R}'$, let $\mathcal{R}_i = \{S_1, S_2, \ldots, S_{i-1}\}$, and $X_i = X' \setminus S_i$. We find the approximate solution Σ_i for this residual instance $(X_i, \mathcal{R}_i, k_i)$ with coverage requirement $k_i = k - |S_i|$, if it is feasible (i.e.

Algorithm 1 PartialCover$(X', \mathcal{R}', k')$.

Sort and renumber the sets in $\mathcal{R}' = \{S_1, \ldots, S_m\}$ such that $w_1 \leq \ldots \leq w_m$.
for $i = 1$ to m **do**
  $\mathcal{R}_i \leftarrow \{S_1, \ldots, S_{i-1}\}$
  $X_i \leftarrow X' \setminus S_i$
  $k_i \leftarrow k' - |S_i|$
  if $(X_i, \mathcal{R}_i, k_i)$ is feasible **then**
    $\Sigma_i \leftarrow$ approximate solution to $(X_i, \mathcal{R}_i, k_i)$
  else
    $\Sigma_i \leftarrow \perp$
  end if
end for
$\ell \leftarrow \arg\min_{i:\Sigma_i \neq \perp} w(\Sigma_i \cup \{S_i\})$
return $\Sigma_\ell \cup \{S_\ell\}$

$\left|\bigcup_{S \in \mathcal{R}_i} S \cap X_i\right| \geq k_i$). We return $\Sigma = \arg\min_{S_i \in \mathcal{R}'} w(\Sigma_i \cup \{S_i\})$ over all S_i such that the residual instance $(X_i, \mathcal{R}_i, k_i)$ is feasible.

▶ **Lemma 1.** *Let Σ^* be an optimal partial cover for the instance $(X', \mathcal{R}', k')$, and let S_p be the heaviest set in Σ^*. Let Σ_p be the approximate solution to $(X_p, \mathcal{R}_p, k_p)$ returned by the Rounding Algorithm of Theorem 3, and Σ' be the solution returned by Algorithm 1. Then,*
1. $OPT(X', \mathcal{R}', k') = OPT(X_p, \mathcal{R}_p, k_p) + w_p$
2. $w(\Sigma') \leq w(\Sigma_p \cup \{S_p\}) \leq (2\beta + 2) \cdot OPT(X', \mathcal{R}', k')$

Proof. Since the optimal solution Σ^* contains S_p, $\Sigma_p^* := \Sigma^* \setminus \{S_p\}$ covers at least $k' - |S_p| = k_p$ elements from $X' \setminus S_p$. Therefore, Σ_p^* is feasible for $(X_p, \mathcal{R}_p, k_p)$. Now, an easy and standard argument implies that Σ_p^* is an optimal solution for $(X_p, \mathcal{R}_p, k_p)$. Thus, $w(\Sigma_p^*) = OPT(X_p, \mathcal{R}_p, k_p)$ and the first part follows.

From Theorem 3, we have an approximate solution Σ_p to the instance $(X_p, \mathcal{R}_p, k_p)$ such that $w(\Sigma_p) \leq (2\beta+2) \cdot OPT(X_p, \mathcal{R}_p, k_p) + B$, where $B \leq w_p$ is the weight of the heaviest set in $\mathcal{R}_p$. Now Algorithm 1 returns a solution whose cost is at most $w(\Sigma_p \cup \{S_p\}) \leq (2\beta + 2) \cdot OPT(X_p, \mathcal{R}_p, k_p) + w_p + w_p \leq (2\beta+2) \cdot (OPT(X_p, \mathcal{R}_p, k_p) + w_p) \leq (2\beta+2) \cdot OPT(X', \mathcal{R}', k')$. We use the result from part 1 in the final inequality. ◀

We summarize our main result in the following theorem, which follows easily from Lemma 1.

▶ **Theorem 2.** *Given our input partial set cover instance $(X', \mathcal{R}', k')$, assume there is a $\beta \geq 1$ such that for any $X_1 \subseteq X'$, we can round a solution to SC LP for the projected set system $(X_1, \mathcal{R}'_{|X_1})$ to within a β factor. Then, we can find a $(2\beta + 2)$-factor approximation for the partial set cover instance $(X', \mathcal{R}', k')$.*

4 Rounding algorithm

Suppose that we have guessed the maximum weight set $S_p \in \mathcal{R}'$ in the optimal solution for the original instance $(X', \mathcal{R}', k')$, as described in the previous section. Thus, we now have the residual instance $(X_p, \mathcal{R}_p, k_p)$, where $X_p = (X' \setminus S_p), \mathcal{R}_p = \{S_1, S_2, \ldots, S_{p-1}\}$, and $k_p = k' - |S_p|$. We solve the LP corresponding to the PSC instance $(X_p, \mathcal{R}_p, k_p)$ to obtain an optimal LP solution $\sigma^* = (x, z)$. In the following, we describe a polynomial time algorithm to round PSC LP on this instance.

Let $0 < \alpha \leq 1/2$ be a parameter (eventually we will set $\alpha = 1/2$). Let $Y = \{e_j \in X_p \mid \sum_{i:e_j \in S_i} x_i \geq \alpha\}$ be the set of elements that are covered to an extent of at least α by the LP solution.

We create a solution σ_1 of a feasible set cover LP for the instance $(Y, \mathcal{R}_{p|Y})$ as follows. For all sets $S_i \in \mathcal{R}_p$, we set $x'_i = \min\{\frac{x_i}{\alpha}, 1\}$. Note that cost of this fractional solution is at most $\frac{1}{\alpha}$ times that of σ^*. Also, note that σ_1 is feasible for the SC LP because for any element $e_j \in Y$, we have that

$$\sum_{i:e_j \in S_i} x'_i = \sum_{i:e_j \in S_i} \min\left\{1, \frac{x_i}{\alpha}\right\} \geq \min\left\{1, \frac{1}{\alpha} \sum_{i:e_j \in S_i} x_i\right\} \geq 1.$$

Suppose that there exists an efficient rounding procedure to round a feasible SC LP solution σ_1, for the instance $(Y, \mathcal{R}_{p|Y})$ to a solution with weight at most $\beta \cdot cost(\sigma_1)$. In the remainder of this section, we describe an algorithm (Algorithm 2) for rounding $\sigma^* = (x, z)$ into a solution that (1) covers at least $k_p - |Y|$ elements from $X_p \setminus Y$, and (2) has cost at most $\frac{1}{\alpha} \cdot cost(\sigma^*) + B$, where B is the weight of the heaviest set in $\mathcal{R}_p$. Combining the two solutions thus acquired, we get the following theorem.

▶ **Theorem 3.** *There exists a rounding algorithm to round a partial cover LP corresponding to $(X_p, \mathcal{R}_p, k_p)$, which returns a solution Σ_p such that $w(\Sigma_p) \leq (2\beta + 2) \cdot OPT(X_p, \mathcal{R}_p, k_p) + B$, where B is the weight of the heaviest set in $\mathcal{R}_p$.*

Proof. Let $\Sigma_p = \Sigma_{p1} \cup \Sigma_{p2}$, where Σ_{p1} is the solution obtained by rounding σ_1, and $\Sigma_{p2} = \Sigma \cup \mathcal{R}_{\text{end}}$ is the solution returned by Algorithm 2. By assumption, Σ_{p1} covers Y, and Σ_{p2} covers at least $k_p - |Y|$ elements from $X_p \setminus Y$ by Lemma 8. Therefore, Σ_p covers at least k_p elements from X_p.

By assumption, we have that $w(\Sigma_{p1}) \leq \beta \cdot cost(\sigma_1) \leq \frac{\beta}{\alpha} cost(\sigma^*)$. Also, from Lemma 9, we have that $w(\Sigma_{p2}) \leq \frac{1}{\alpha} cost(\sigma^*) + B$. We get the claimed result by combining the previous two inequalities, setting $\alpha = 1/2$, and noting that $cost(\sigma^*) \leq OPT(X_p, \mathcal{R}_p, k_p)$. ◀

Let $\mathcal{H} = \{S_i \in \mathcal{R}_p \mid x_i \geq \alpha\}$ be the sets that have x_i value at least α. Note that without loss of generality, we can assume that $\cup_{S_i \in \mathcal{H}} S_i \subseteq Y$. If $|Y| \geq k_p$, we are done. Otherwise, let $X \leftarrow X_p \setminus Y$, $\mathcal{R} \leftarrow \mathcal{R}_p \setminus \mathcal{H}$, and $k \leftarrow k_p - |Y|$. Let $\sigma = (x, z)$ be the LP solution σ^* restricted to the instance $(X, \mathcal{R}, k)$, that is, $x = (x_i \mid S_i \in \mathcal{R}), z = (z_j \mid e_j \in X)$. We show how to round σ on the instance $(X, \mathcal{R}, k)$ to find a collection of sets that covers at least k elements from X. In the following lemma, we show that the LP solution σ is feasible for the instance $(X, \mathcal{R}, k)$.

▶ **Lemma 4.** *The LP solution $\sigma = (x, z)$ is feasible for the instance $(X, \mathcal{R}, k)$. Furthermore, $cost(\sigma) \leq cost(\sigma^*)$.*

Proof. Note that x_i and z_j values are unchanged from the optimal solution σ^*, therefore the Constraints 9, and 8 (from PSC LP) are satisfied.

Note that by definition, for any element $e_j \in X$, $e_j \notin \cup_{S_{i'} \in \mathcal{H}} S_{i'}$, and $e_j \notin Y$. Therefore, by Constraint 6, we have that $\sum_{i:e_j \in S_i} x_i = \sum_{i:e_j \in S_i, S_i \in \mathcal{R}} x_i \geq z_j$.

As for Constraint 7, note that

$$\begin{aligned}
&\sum_{e_j \in X_p} z_j \geq k_p && \text{(By feasibility of optimal solution } \sigma^*\text{)} \\
\implies &\sum_{e_j \in X} z_j \geq k_p - \sum_{e_j \in Y} z_j && (X = X_p \setminus Y) \\
\implies &\sum_{e_j \in X} z_j \geq k_p - |Y| && (z_j \leq 1 \text{ for } e_j \in Y \text{ by feasibility}) \\
\implies &\sum_{e_j \in X} z_j \geq k && (k = k_p - |Y|)
\end{aligned}$$

Finally, note that $cost(\sigma) = \sum_{S_i \in \mathcal{R}} w_i x_i \leq \sum_{S_i \in \mathcal{R}_p} w_i x_i = cost(\sigma^*)$, because $\mathcal{R} \subseteq \mathcal{R}_p$, and the x_i values are unchanged. ◀

4.1 Algorithm for rounding shallow elements

We have an LP solution σ for the PSC instance $(X, \mathcal{R}, k)$. Note that for any $S_i \in \mathcal{R}$, $x_i < \alpha$, and for any $e_j \in X, \alpha > \sum_{i:e_j \in S_i} x_i \geq z_j$, i.e. each element is *shallow*. For convenience, we let $z_j = \sum_{i:e_j \in S_i} x_i$. We now describe Algorithm 2, which rounds σ to an integral solution to the instance $(X, \mathcal{R}, k)$. At the beginning of Algorithm 2, we initialize $\mathcal{R}_{\text{cur}}$, the collection of "unresolved" sets, to be $\mathcal{R}$; and X_{cur}, the set of "uncovered" elements, to be X.

At the heart of the rounding algorithm is the procedure RoundTwoSets, which takes input two sets $S_1, S_2 \in \mathcal{R}_{\text{cur}}$, and rounds the corresponding variables x_1, x_2 such that either x_1 is increased to α, or x_2 is decreased to 0 (cf. Lemma 5 part 3). A set is removed from $\mathcal{R}_{\text{cur}}$ if either of these conditions is met. In addition, if x_i reaches α, then the set S_i is added to Σ, which is a part of the output, and all the elements in S_i are added to the set Ξ; furthermore, x_i is set to 1. At a high level, the goal of Algorithm 2 is to resolve all of the sets either way, while maintaining the cost and the feasibility of the LP.

Given the procedure RoundTwoSets, we carefully choose the order in which the sets are paired up for rounding; however, there is some degree of freedom. We pick a set from $\mathcal{R}_{\text{cur}}$ in a careful way as the *leader*. We use variable a to denote the index of the leader; thus, the leader is S_a. The leader S_a is chosen arbitrarily in Line 4 but in a specific way in Line 20. We keep pairing the leader S_a up with another arbitrary set $S_b \in \mathcal{R}_{\text{cur}}$, until S_a is removed from $\mathcal{R}_{\text{cur}}$, or it is the only set remaining in $\mathcal{R}_{\text{cur}}$. To ensure that the Constraint 7 is maintained, we carefully determine whether to increase x_a and decrease x_b in RoundTwoSets, or vice versa. Thinking of $\frac{|X_{\text{cur}} \cap S_a|}{w_a}$, and $\frac{|X_{\text{cur}} \cap S_b|}{w_b}$ as the "cost-effectiveness" of the sets S_a and S_b respectively, we increase x_a at the expense of x_b, if S_a is more cost-effective than S_b or vice versa.

Notice that, instead of fixing a set S_a and pairing it up with other sets S_b, if we arbitrarily chose the pairs of sets to be rounded, then the feasibility of the LP may not be maintained. In particular, we cannot ensure that for all elements $e_j \in X_{\text{cur}}$, $z_j \leq 1$ (Constraint 8). To this end, we show that, our order of pairing up sets maintains the following two invariants:

1. Let $X_\alpha = \{e_j \in X_{\text{cur}} \mid z_j \geq \alpha\}$. During the execution of while loop of Line 3, the elements of X_α are contained in the leader $S_a \in \mathcal{R}_{\text{cur}}$, that is chosen in Line 4 or Line 20.
2. Fix any set $S_i \in \mathcal{R}_{\text{cur}} \setminus \{S_a\}$. The x_i value is unchanged since the beginning of the algorithm until the beginning of the current iteration of while loop of Line 5; the x_i value can change in the current iteration only if S_i is paired up with S_a.

In Lemma 7, we show that these invariants imply that Constraint 8 is maintained.

Algorithm 2 RoundLP$(X, \mathcal{R}, w, k, \sigma)$.

1: $\Sigma \leftarrow \emptyset$, $\Xi \leftarrow \emptyset$.
2: $X_{\text{cur}} \leftarrow X$, $\mathcal{R}_{\text{cur}} \leftarrow \mathcal{R}$.
3: **while** $|\mathcal{R}_{\text{cur}}| \geq 2$ **do**
4: $a \leftarrow \ell$, where S_ℓ is an arbitrary set from $\mathcal{R}_{\text{cur}}$.
5: **while** $0 < x_a < \alpha$ and $|\mathcal{R}_{\text{cur}} \setminus \{S_a\}| \geq 1$ **do**
6: $S_b \leftarrow$ an arbitrary set from $\mathcal{R}_{\text{cur}} \setminus \{S_a\}$.
7: **if** $\frac{|X_{\text{cur}} \cap S_a|}{w_a} \geq \frac{|X_{\text{cur}} \cap S_b|}{w_b}$ **then**
8: $(x_a, x_b, z) \leftarrow$RoundTwoSets$(S_a, S_b, w, \sigma, X_{\text{cur}}, \mathcal{R}_{\text{cur}})$
9: **if** $x_b = 0$ **then**
10: $\mathcal{R}_{\text{cur}} \leftarrow \mathcal{R}_{\text{cur}} \setminus \{S_b\}$.
11: **end if**
12: **if** $x_a = \alpha$ **then**
13: $\Xi \leftarrow \Xi \cup S_a$, $X_{\text{cur}} \leftarrow X_{\text{cur}} \setminus S_a$.
14: $\Sigma \leftarrow \Sigma \cup \{S_a\}, \mathcal{R}_{\text{cur}} \leftarrow \mathcal{R}_{\text{cur}} \setminus \{S_a\}$, $x_a \leftarrow 1$.
15: **end if**
16: **else**
17: $(x_b, x_a, z) \leftarrow$RoundTwoSets$(S_b, S_a, w, \sigma, X_{\text{cur}}, \mathcal{R}_{\text{cur}})$.
18: **if** $x_a = 0$ **then**
19: $\mathcal{R}_{\text{cur}} \leftarrow \mathcal{R}_{\text{cur}} \setminus \{S_a\}$.
20: $a \leftarrow b$.
21: **end if**
22: **if** $x_b = \alpha$ **then**
23: $\Xi \leftarrow \Xi \cup S_b$, $X_{\text{cur}} \leftarrow X_{\text{cur}} \setminus S_b$.
24: $\Sigma \leftarrow \Sigma \cup \{S_b\}, \mathcal{R}_{\text{cur}} \leftarrow \mathcal{R}_{\text{cur}} \setminus \{S_b\}$, $x_b \leftarrow 1$.
25: **end if**
26: **end if**
27: **end while**
28: **end while**
29: $\mathcal{R}_{\text{end}} \leftarrow \mathcal{R}_{\text{cur}}$.
30: **return** $\Sigma \cup \mathcal{R}_{\text{end}}$.

31: **function** RoundTwoSets$(S_1, S_2, w, \sigma, X_{\text{cur}}, \mathcal{R}_{\text{cur}})$
32: $\delta \leftarrow \min\{\alpha - x_1, \frac{w_2}{w_1} \cdot x_2\}$.
33: $x_1 \leftarrow x_1 + \delta$.
34: $x_2 \leftarrow x_2 - \frac{w_1}{w_2} \cdot \delta$.
35: For all elements $e_j \in X_{\text{cur}}$, update $z_j \leftarrow \sum_{i:e_j \in S_i} x_i$.
36: **return** (x_1, x_2, z).
37: **end function**

The invariants are trivially true at the start of the first iteration of the while loops. Let $S_a \in \mathcal{R}_{\text{cur}}$ be a set chosen in Line 4, or Line 20. During the while loop, we maintain the invariants by pairing up the S_a with other arbitrary sets S_b, until S_a is removed from $\mathcal{R}_{\text{cur}}$ in one of the two ways; or until it is the last set remaining. It is easy to see that Invariant 2 is maintained.

Now we describe in detail how Invariant 1 is being maintained in the course of the algorithm. Consider the first case, i.e. in RoundTwoSets, we increase x_a and decrease x_b. If after this, x_b becomes 0, then we remove S_b from $\mathcal{R}_{\text{cur}}$. If, on the other hand, x_a increases to α, then all the elements in $X_{\text{cur}} \cap S_a$ are covered to an extent of at least α, and so we remove S_a from $\mathcal{R}_{\text{cur}}$ and $S_a \cap X_{\text{cur}}$ from X_{cur}. If x_b becomes 0, the set X_α continues to be a subset of S_a, and if x_a increases to α, it becomes empty. Thus, Invariant 1 is maintained.

In the second case, in RoundTwoSets, x_a is decreased and x_b is increased. This case is a bit more complicated, because z_j values of elements $e_j \in S_b$ are being increased by virtue of increase in x_b. Therefore, we need to explicitly maintain Invariant 1. If x_b reaches α, then S_b is removed from $\mathcal{R}_{\text{cur}}$ and all the elements covered by S_b are removed from X_{cur} (and thus the invariant is maintained). On the other hand, if x_a reaches 0, then the net change in the z_j values (since the beginning of the algorithm) for the elements $e_j \in S_a \setminus S_b$ is non-positive – this follows from Invariant 2, as the x_i values of the sets in $\mathcal{R}_{\text{cur}} \setminus \{S_a, S_b\}$ are unchanged, and x_a is now zero. Therefore, the set $X_\alpha \cap (S_a \setminus S_b) = \varnothing$. However, $X_\alpha \cap S_b$ may be non-empty because of the increase in x_b. Therefore, we reset a to b, thus obtaining a new leader $S_a = S_b$, and continue pairing the new leader up with other sets. Notice that we have maintained Invariant 1 although the leader S_a has changed.

From the above discussion, we have the following result.

▶ **Claim 1.** *Throughout the execution of the while loop of Line 3, Invariants 1, and 2 are maintained.*

Finally, after leaving the while loop of Line 3, we set $\mathcal{R}_{\text{end}}$ to be $\mathcal{R}_{\text{cur}}$, and add it to our solution. Note that at this point, $\mathcal{R}_{\text{cur}}$ contains at most one set. We show that the resulting solution $\Sigma \cup \mathcal{R}_{\text{end}}$ covers at least k elements.

4.2 Analysis

In this section, we analyze the behavior of Algorithm 2. In the following lemma, we show that in each iteration, we make progress towards rounding while maintaining the cost of the LP solution.

▶ **Lemma 5.** *Let $\sigma = (x, z), \sigma' = (x', z')$ be the LP solutions just before and after the execution of* RoundTwoSets$(S_1, S_2, w, \sigma, X_{cur}, \mathcal{R}_{cur})$ *for some sets $S_1, S_2 \in \mathcal{R}_{cur}$ in some iteration of the algorithm, such that σ is a feasible solution to the LP. Then,*

1. *$cost(\sigma) = cost(\sigma')$.*
2. *$\sum_{e_j \in X_{cur}} z'_j \geq \sum_{e_j \in X_{cur}} z_j$.*
3. *Either $x'_1 = \alpha$ or $x'_2 = 0$ (or both).*

Proof.

1. Note that the x_i variables corresponding to all the sets $S_i \notin \{S_1, S_2\}$ remain unchanged. The net change in the cost of the LP solution is
$$w_1 \cdot (x'_1 - x_1) + w_2 \cdot (x'_2 - x_2) = w_1 \cdot \delta - w_2 \cdot \left(\frac{w_1}{w_2} \cdot \delta\right) = 0.$$
2. Let $A = S_1 \cap X_{\text{cur}}$, and $B = S_2 \cap X_{\text{cur}}$. $z'_j = z_j$ for all elements $e_j \notin A \cup B$, i.e. z_j values are modified only for the elements $e_j \in A \cup B$.
For $|A|$ elements $e_j \in A$, z_j value is increased by δ by virtue of increase in x_1. Similarly, for $|B|$ elements $e_{j'} \in B$, $z_{j'}$ value is decreased by $\frac{w_1}{w_2} \cdot \delta$. However by assumption, we have that $\frac{|A|}{w_1} \geq \frac{|B|}{w_2}$. Therefore, the net change in the sum of z_j values is
$$|A| \cdot \delta - |B| \cdot \left(\frac{w_1}{w_2} \cdot \delta\right) \geq |A| \cdot \delta - \left(\frac{|A|}{w_1} \cdot w_1\right) \cdot \delta \geq 0.$$
3. The value of δ is chosen such that $\delta = \min\{\alpha - x_1, \frac{w_2}{w_1} \cdot x_2\}$. If $\delta = \alpha - x_1 \leq \frac{w_2}{w_1} \cdot x_2$, then $x'_1 = x_1 + (\alpha - x_1) = \alpha$, and $x'_2 = x_2 - \frac{w_1}{w_2} \cdot (\alpha - x_1) \geq x_2 - x_2 = 0$. In the other case when $\delta = \frac{w_2}{w_1} \cdot x_2 < (\alpha - x_1)$, we have that $x'_1 = x_1 + \frac{w_2}{w_1} \cdot x_2 < x_1 + (\alpha - x_1) = \alpha$, and $x'_2 = x_2 - \frac{w_2}{w_1} \cdot \frac{w_1}{w_2} \cdot x_2 = 0$. ◀

▶ Remark. Note that Lemma 5 (in particular, the Part 2 of Lemma 5) alone is not sufficient to show the feasibility of the LP after an execution of RoundTwoSets – we also have to show that $z'_j \leq 1$. This is slightly involved, and is shown in Lemma 7 with the help of Invariants 1, and 2.

▶ **Corollary 6.** *Algorithm 2 runs in polynomial time.*

Proof. In each iteration of the inner while loop Line 5, RoundTwoSets is called on some two sets $S_1, S_2 \in \mathcal{R}_{\text{cur}}$, and as such from Lemma 5, either $x'_1 = \alpha$ or $x'_2 = 0$. Therefore, at least one of the sets is removed from $\mathcal{R}_{\text{cur}}$ in each iteration. Therefore, there are at most $O(|\mathcal{R}|)$ iterations of the inner while loop. It is easy to see that each execution of RoundTwoSets takes $O(|\mathcal{R}| \cdot |X|)$ time. ◀

In the following Lemma, we show that Constraints 8 is being maintained by the algorithm. This, when combined with Lemma 5, shows that we maintain the feasibility of the LP at all times.

▶ **Lemma 7.** *During the execution of Algorithm 2, for any element $e_j \in X_{cur}$, we have that $z_j \leq 2\alpha$. By the choice of range of α, the feasibility of the LP is maintained.*

Proof. At the beginning of the algorithm, we have that $z_j \leq \alpha$ for all elements $e_j \in X_{\text{cur}} = X$. Now at any point in the while loop, consider the set $X_\alpha = \{e_j \in X_{\text{cur}} \mid z_j \geq \alpha\}$ as defined earlier. For any element $e_j \in X_{\text{cur}} \setminus X_\alpha$, the condition is already met, therefore we need to argue only for the elements in X_α. We know by Invariant 1 that there exists a set $S_a \in \mathcal{R}_{\text{cur}}$ such that $X_\alpha \subseteq S_a$.

By Invariant 2, the x_i values of all sets $S_i \in \mathcal{R}_{\text{cur}} \setminus \{S_a\}$ are unchanged, and therefore for all elements $e_j \in X_{\text{cur}}$, the net change to the z_j variable is positive only by the virtue of increase in the x_a value. However, the net increase in the x_a value is at most α because a set S_i is removed from $\mathcal{R}_{\text{cur}}$ as soon as its x_i value reaches α. Accounting for the initial z_j value which is at most α, we conclude that $z_j \leq 2\alpha$. ◀

Note that after leaving the outer while loop (Line 28), we must have $|\mathcal{R}_{\text{cur}}| \leq 1$. That is in Line 29, we either let $\mathcal{R}_{\text{end}} \leftarrow \mathcal{R}_{\text{cur}} = \emptyset$, or $\mathcal{R}_{\text{end}} \leftarrow \mathcal{R}_{\text{cur}} = \{S_i\}$ for some set $S_i \in \mathcal{R}$.

To state the following claim, we introduce the following notation. Let $\sigma' = (x', z')$ be the LP solution at the end of Algorithm 2. Let $\mathcal{R}_r = \mathcal{R} \setminus \Sigma$, where Σ is the collection at the end of the while loop of Algorithm 2, and let $X_r = X \setminus \Xi$. Note that any element $e_j \in X_r$ is contained only in the sets of $\mathcal{R}_r$. Finally, let $Z_r = \sum_{e_j \in X_r} z'_j$.

▶ **Claim 2.** *If $\mathcal{R}_{end} \neq \emptyset$, then at least Z_r elements are covered by $\mathcal{R}_{end}$.*

Proof. By assumption, we have that $\mathcal{R}_{\text{end}} \neq \emptyset$, i.e. $\mathcal{R}_{\text{end}} = \{S_i\}$ for some $S_i \in \mathcal{R}$. For each $S_l \in \mathcal{R}_r$ with $l \neq i$, we have that $x_l = 0$, again by the condition of the outer while loop. Since Constraint 6 is made tight for all elements in each execution of RoundTwoSets, for any element $e_{j'} \in X_r$ but $e_{j'} \notin S_i$, we have that $z_{j'} = 0$. On the other hand, for elements $e_j \in X_r \cap S_i$, we have that $z'_j = x'_i \leq \alpha$. If the number of such elements is p, then we have that $Z_r \leq \alpha \cdot p$. The lemma follows since choosing S_i covers all of these p elements, and $p \geq Z_r/\alpha \geq Z_r$. ◀

In the following lemma, we show that Algorithm 2 produces a feasible solution.

▶ **Lemma 8.** *The solution $\Sigma \cup \mathcal{R}_{end}$ returned by Algorithm 2 covers at least k elements.*

Proof. There are two cases – $\mathcal{R}_{\text{end}} = \emptyset$, or $\mathcal{R}_{\text{end}} = \{S_i\}$ for some $S_i \in \mathcal{R}$. In the first case, all elements in X_r are uncovered, and for all such elements, $e_{j'} \in X_r$, we have that $z_{j'} = 0$. In this case, it is trivially true that the number of elements of X_r covered by $\mathcal{R}_{\text{end}}$ is Z_r $(= 0)$. In the second case, the same follows from Claim 2. Therefore, in both cases we have that,

$$\begin{aligned}
\text{Number of elements covered} &\geq |\Xi| + Z_r \\
&\geq \sum_{e_j \in \Xi} z'_j + \sum_{e_j \in X_r} z'_j && \text{(By Lemma 7 and } z'_j \leq 1\text{)} \\
&= \sum_{e_j \in X} z'_j \\
&\geq \sum_{e_j \in X} z_j && \text{(Lemma 5, Part 2)} \\
&\geq k && \text{(By Lemma 4 and Constraint 7)}
\end{aligned}$$

Recall that z_j refers to the z-value of an element e_j in the optimal LP solution σ, at the beginning of the algorithm. ◀

▶ **Lemma 9.** *Let $\Sigma \cup \mathcal{R}_{end}$ be the solution returned by Algorithm 2, and let B be the weight of the heaviest set in $\mathcal{R}$. Let $\sigma = (x, z)$ and $\sigma' = (x', z')$ denote the LP solutions at the beginning and end of Algorithm 2, respectively. Then,*

1. $w(\Sigma) = \sum_{S_i \in \Sigma} w_i x'_i \leq \sum_{S_i \in \mathcal{R}} w_i x'_i \leq \frac{1}{\alpha} \sum_{S_i \in \mathcal{R}} w_i x_i = \frac{1}{\alpha} cost(\sigma)$,
2. $w(\mathcal{R}_{end}) \leq B$, *and*
3. $w(\Sigma \cup \mathcal{R}_{end}) \leq \frac{1}{\alpha} cost(\sigma) + B$.

Proof. For the first part, the inequality $\sum_{S_i \in \mathcal{R}} w_i x'_i \leq \frac{1}{\alpha} \sum_{S_i \in \mathcal{R}} w_i x_i$ follows because (a) RoundTwoSets preserves the cost of the LP solution, and (b) when a set S_i is added to Σ, its contribution to the cost of the LP increases by a factor of $\frac{1}{\alpha}$. For the second part, note that $\mathcal{R}_{\text{end}}$ contains at most one set $S_i \in \mathcal{R}$. By definition, weight of any set in S_i is bounded by B, the maximum weight of any set in $\mathcal{R}$. The third part follows from the first two parts. ◀

From Lemma 8 and Lemma 9, we conclude that $\Sigma \cup \mathcal{R}_{\text{end}}$ is a solution that covers at least $k = k_p - |X_1|$ elements from $X_p \setminus X_1$, and whose cost is at most $\frac{1}{\alpha} cost(\sigma) + B$.

5 A generalization of PSC

Consider the following generalization of the PSC problem, where the elements $e_j \in X'$ have profits $p_j \geq 0$ associated with them. Now the goal is to choose a minimum-weight collection $\Sigma \subseteq \mathcal{R}'$ such that the total profit of elements covered by the sets of Σ is at least K, where $0 \leq K \leq \sum_{e_j \in X} p_j$ is provided as an input. Note that setting $p_j = 1$ for all elements we get the original PSC problem. This generalization has been considered in [26].

It is easy to modify our algorithm that for PSC, such that it returns a $2\beta + 2$ approximate solution for this generalization as well. We briefly describe the modifications required. Firstly, we modify Constraint 7 of PSC LP to incorporate the profits as follows:

$$\sum_{e_j \in X} z_j \cdot p_j \geq K$$

The preprocessing and the rounding algorithms work with the straightforward modifications required to handle the profits. One significant change is in the rounding algorithm (Algorithm

2). We compare the "cost-effectiveness" of the two sets S_a, S_b in Line 7 for the PSC as $\frac{|S_a \cap X_{\text{cur}}|}{w_a} \geq \frac{|S_b \cap X_{\text{cur}}|}{w_b}$. For handling the profits of the elements, we replace this with the following condition – $\frac{P_a}{w_a} \geq \frac{P_b}{w_b}$, where $P_a := \sum_{e_j \in S_a \cap X_{\text{cur}}} p_j$, and $P_b := \sum_{e_j \in S_b \cap X_{\text{cur}}} p_j$. With similar straightforward modifications, the analysis of Algorithm 2 goes through with the same guarantee on the cost of the solution. We remark here that despite the profits, the approximation ratio only depends on that of the standard SC LP, which is oblivious to the profits.

References

1. 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.
2. Reuven Bar-Yehuda. Using homogeneous weights for approximating the partial cover problem. *Journal of Algorithms*, 39(2):137–144, 2001.
3. Hervé Brönnimann and Michael T. Goodrich. Almost optimal set covers in finite vc-dimension. *Discrete & Computational Geometry*, 14(4):463–479, 1995.
4. Nader H. Bshouty and Lynn Burroughs. Massaging a linear programming solution to give a 2-approximation for a generalization of the vertex cover problem. In *STACS 98, 15th Annual Symposium on Theoretical Aspects of Computer Science, Paris, France, February 25-27, 1998, Proceedings*, pages 298–308, 1998.
5. Timothy M. Chan. Polynomial-time approximation schemes for packing and piercing fat objects. *J. Algorithms*, 46(2):178–189, 2003.
6. Timothy M. Chan, Elyot Grant, Jochen Könemann, and Malcolm Sharpe. Weighted capacitated, priority, and geometric set cover via improved quasi-uniform sampling. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 1576–1585, 2012.
7. Timothy M. Chan and Nan Hu. Geometric red-blue set cover for unit squares and related problems. *Comput. Geom.*, 48(5):380–385, 2015.
8. Kenneth L. Clarkson. New applications of random sampling in computational geometry. *Discrete Comput. Geom.*, 2(1):195–222, 1987.
9. Kenneth L. Clarkson and Kasturi Varadarajan. Improved approximation algorithms for geometric set cover. *Discrete & Computational Geometry*, 37(1):43–58, 2007.
10. Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *Proceedings of the Forty-sixth Annual ACM Symposium on Theory of Computing*, STOC '14, pages 624–633, New York, NY, USA, 2014. ACM.
11. Khaled M. Elbassioni, Erik Krohn, Domagoj Matijevic, Julián Mestre, and Domagoj Severdija. Improved approximations for guarding 1.5-dimensional terrains. *Algorithmica*, 60(2):451–463, 2011.
12. Thomas Erlebach and Erik Jan Van Leeuwen. Ptas for weighted set cover on unit squares. In *Proceedings of the 13th International Conference on Approximation, and 14 the International Conference on Randomization, and Combinatorial Optimization: Algorithms and Techniques*, APPROX/RANDOM'10, pages 166–177, Berlin, Heidelberg, 2010. Springer-Verlag.
13. Guy Even, Dror Rawitz, and Shimon (Moni) Shahar. Hitting sets when the vc-dimension is small. *Inf. Process. Lett.*, 95(2):358–362, jul 2005.
14. Esther Ezra, Boris Aronov, and Micha Sharir. Improved bound for the union of fat triangles. In *Proceedings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '11, pages 1778–1785, Philadelphia, PA, USA, 2011. Society for Industrial and Applied Mathematics.
15. Uriel Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652, 1998.

16 Toshihiro Fujito. On combinatorial approximation of covering 0-1 integer programs and partial set cover. *Journal of Combinatorial Optimization*, 8(4):439–452, 2001.

17 Rajiv Gandhi, Samir Khuller, and Srinivasan Aravind. Approximation algorithms for partial covering problems. *J. Algorithms*, 53(1):55–84, 2004.

18 Christian Glaßer, Christian Reitwießner, and Heinz Schmitz. Multiobjective disk cover admits a PTAS. In *Algorithms and Computation, 19th International Symposium, ISAAC 2008, Gold Coast, Australia, December 15-17, 2008. Proceedings*, pages 40–51, 2008.

19 Sathish Govindarajan, Rajiv Raman, Saurabh Ray, and Aniket Basu Roy. Packing and covering with non-piercing regions. In *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*, pages 47:1–47:17, 2016.

20 David Haussler and Emo Welzl. epsilon-nets and simplex range queries. *Discrete & Computational Geometry*, 2:127–151, 1987.

21 Dorit S. Hochbaum. The t-vertex cover problem: Extending the half integrality framework with budget constraints. In *Proceedings of the International Workshop on Approximation Algorithms for Combinatorial Optimization*, APPROX '98, pages 111–122, London, UK, UK, 1998. Springer-Verlag.

22 Dorit S. Hochbaum and Wolfgang Maass. Approximation schemes for covering and packing problems in image processing and vlsi. *J. ACM*, 32(1):130–136, 1985.

23 Michael J. Kearns. *Computational Complexity of Machine Learning*. MIT Press, Cambridge, MA, USA, 1990.

24 Samir Khuller, Anna Moss, and Joseph Naor. The budgeted maximum coverage problem. *Inf. Process. Lett.*, 70(1):39–45, 1999.

25 James King and David Kirkpatrick. Improved approximation for guarding simple galleries from the perimeter. *Discrete Comput. Geom.*, 46(2):252–269, 2011.

26 Jochen Könemann, Ojas Parekh, and Danny Segev. A unified approach to approximating partial covering problems. *Algorithmica*, 59(4):489–509, 2011.

27 Erik Krohn, Matt Gibson, Gaurav Kanade, and Kasturi R. Varadarajan. Guarding terrains via local search. *JoCG*, 5(1):168–178, 2014.

28 Julián Mestre. A primal-dual approximation algorithm for partial vertex cover: Making educated guesses. *Algorithmica*, 55(1):227–239, 2009.

29 Nabil H. Mustafa and Saurabh Ray. Improved results on geometric hitting set problems. *Discrete & Computational Geometry*, 44(4):883–895, 2010.

30 Petr Slavík. Improved performance of the greedy algorithm for partial cover. *Inf. Process. Lett.*, 64(5):251–254, dec 1997.

31 Kasturi R. Varadarajan. Epsilon nets and union complexity. In *Proceedings of the 25th ACM Symposium on Computational Geometry, Aarhus, Denmark, June 8-10, 2009*, pages 11–16, 2009.

32 Kasturi R. Varadarajan. Weighted geometric set cover via quasi-uniform sampling. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 641–648, 2010.

33 Vijay V. Vazirani. *Approximation Algorithms*. Springer-Verlag New York, Inc., New York, NY, USA, 2001.

Optimality of Geometric Local Search

Bruno Jartoux
Laboratoire d'Informatique Gaspard-Monge, Université Paris-Est, ESIEE Paris
Marne-la-Vallée, France
bruno.jartoux@esiee.fr
https://orcid.org/0000-0002-5341-1968

Nabil H. Mustafa
Laboratoire d'Informatique Gaspard-Monge, Université Paris-Est, ESIEE Paris
Marne-la-Vallée, France
mustafan@esiee.fr

Abstract

Up until a decade ago, the algorithmic status of several basic NP-complete problems in geometric combinatorial optimisation was unresolved. This included the existence of polynomial-time approximation schemes (PTASs) for hitting set, set cover, dominating set, independent set, and other problems for some basic geometric objects. These past nine years have seen the resolution of all these problems—interestingly, with the same algorithm: local search. In fact, it was shown that for many of these problems, local search with radius λ gives a $(1+O(\lambda^{-\frac{1}{2}}))$-approximation with running time $n^{O(\lambda)}$. Setting $\lambda = \Theta(\epsilon^{-2})$ yields a PTAS with a running time of $n^{O(\epsilon^{-2})}$.

On the other hand, hardness results suggest that there do not exist PTASs for these problems with running time $\text{poly}(n) \cdot f(\epsilon)$ for any arbitrary f. Thus the main question left open in previous work is in improving the exponent of n to $o(\epsilon^{-2})$.

We show that in fact the approximation guarantee of local search cannot be improved for any of these problems. The key ingredient, of independent interest, is a new lower bound on locally expanding planar graphs, which is then used to show the impossibility results. Our construction extends to other graph families with small separators.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis, Mathematics of computing → Combinatorial optimization, Mathematics of computing → Graph theory, Mathematics of computing → Approximation algorithms

Keywords and phrases local search, expansion, matchings, Hall's marriage theorem

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.48

Funding This work was supported by ANR grant SAGA (JCJC-14-CE25-0016-01).

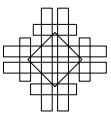
Acknowledgements We thank the referees for several helpful comments.

1 Introduction

Within the past decade polynomial-time approximation schemes (PTASs) have been proposed for a number of long-standing open problems in geometric approximation algorithms, including the following NP-hard problems (see [18, 10] for hardness results):

Hitting set for pseudodisks [23]: given a set $\mathcal{P}$ of points and a family $\mathcal{D}$ of pseudodisks[1] in the plane, compute a smallest subset of $\mathcal{P}$ that intersects all pseudodisks in $\mathcal{D}$.

[1] A **family of pseudodisks** is a set of planar regions whose boundaries are Jordan curves and such that the boundaries of any pair of pseudodisks intersect at most twice.



Independent set of pseudodisks [1, 9]: given a family $\mathcal{D}$ of pseudodisks, compute a maximum size subset of pairwise disjoint pseudodisks in $\mathcal{D}$.

Dominating set of pseudodisks [14, 15]: given a family $\mathcal{D}$ of pseudodisks, compute a smallest subset of pseudodisks of $\mathcal{D}$ that together intersect all other pseudodisks of $\mathcal{D}$.

Set cover for disks [8, 5]: given a set $\mathcal{P}$ of points and a family $\mathcal{D}$ of disks in the plane, return a smallest subset of disks in $\mathcal{D}$ that together cover all points of $\mathcal{P}$.

Unit-capacity point-packing [12]: given a set of points $\mathcal{P}$ and a set of disks $\mathcal{D}$, compute a largest subset of $\mathcal{P}$ that hits no disk of $\mathcal{D}$ more than once.

1.1 Local search

Surprisingly, the PTAS for all these problems is essentially the same: **local search**. Let X be the set of base elements of the problem, and let the **search radius** $\lambda \geq 3$ be an integer. Then start with any feasible solution $\mathcal{S} \subseteq X$ and increase (in the case of a maximisation problem, e.g., maximum independent set) or decrease (in the case of a minimisation problem, e.g., minimum hitting set) its size by local improvement steps while maintaining feasibility. Here a **local improvement step** is to swap a subset $\mathcal{S}'$ of at most λ elements of the current solution $\mathcal{S}$ with a subset of $X \setminus \mathcal{S}$ of size at least $|\mathcal{S}'| + 1$ (for maximisation problems) or at most $|\mathcal{S}'| - 1$ (for minimisation problems), as long as the new solution is still feasible. The algorithm finishes when no local improvement step is possible. Such a solution is called **λ-locally optimal**.

All these algorithms are analysed in a similar way, as follows. Let $\mathcal{S}$ be a λ-locally optimal solution and $\mathcal{O}$ be an optimal solution[2]. To relate the cardinalities of $\mathcal{S}$ and $\mathcal{O}$, a bipartite **exchange graph** is built on vertex sets $\mathcal{S}$ and $\mathcal{O}$ with a local vertex expansion property[3]:

Minimisation: for all $\mathcal{S}' \subseteq \mathcal{S}$ of size at most λ, $|N(\mathcal{S}')| \geq |\mathcal{S}'|$. (1)

Maximisation: for all $\mathcal{O}' \subseteq \mathcal{O}$ of size at most λ, $|N(\mathcal{O}')| \geq |\mathcal{O}'|$. (2)

The construction of exchange graphs is problem-specific and exploits the geometric properties of optimal and local solutions. For example, in the minimum vertex cover problem on a graph G this would simply be the bipartite subgraph of G induced by $\mathcal{S}$ and $\mathcal{O}$; condition (1) follows from the local optimality of $\mathcal{S}$.

The key in the analysis lies in a general theorem on local expansion in sparse graphs. A bipartite graph on vertex sets (B, R) is **λ-expanding**[4] if for all $B' \subseteq B$ of size at most λ we have $|N(B')| \geq |B'|$. A **(vertex) separator** of a graph on n vertices is a subset of vertices whose removal leaves connected components of cardinality at most $\frac{2}{3}n$. A class of graphs $\mathcal{G}$ has the **separator property** with parameter $s \in [0, 1]$ if there exists a positive constant c such that any graph in $\mathcal{G}$ has a separator of size at most cn^{1-s}, where n is the number of vertices. For example, trees have this property with $s = 1$ as they have constant-sized separators, whereas planar graphs have the separator property with parameter $s = \frac{1}{2}$. In fact, the separator property with $s = \frac{1}{2}$ actually holds for graphs excluding fixed minors and in particular for minor-closed classes other than the class of all graphs, e.g. graphs of bounded genus [2]. A class of graphs closed under taking subgraphs is called **monotone**.

---

[2] We can assume that these solutions are disjoint by considering $\mathcal{S} \setminus \mathcal{O}$ and $\mathcal{O} \setminus \mathcal{S}$.

[3] For a graph $G = (V, E)$ (which will always be clear from the context) and a set $V' \subseteq V$, $N(V')$ denotes the set of neighbours of the vertices of V' in G.

[4] Note that the roles of B and R are *not* symmetric.

▶ **Theorem 1** ([23, 9, 5]). *If a finite and λ-expanding bipartite graph on (B, R) belongs to a monotone family with the separator property with parameter $s \in (0, 1)$ and $\lambda \geq \lambda_s$, then $|B| \leq (1 + c_s\lambda^{-s}) \cdot |R|$, where c_s and λ_s are positive constants that depend only on s.*

In an independent paper, Cabello and Gajser [7] describe a subcase of this theorem for K_h-minor-free graphs, which have separators of size $O(h\sqrt{n})$. Finally, Har-Peled and Quanrud [16, 17] observe that intersection graphs of low-density objects in $\mathbb{R}^d$ have the separator property with $s = \frac{1}{d}$.

To complete the analysis for minimisation problems, apply Theorem 1 with $B = \mathcal{S}$ and $R = \mathcal{O}$, and get $|\mathcal{S}| \leq (1 + c_s\lambda^{-s}) \cdot |\mathcal{O}|$. For maximisation problems, take $B = \mathcal{O}$ and $R = \mathcal{S}$, and get $|\mathcal{O}| \leq (1 + c_s\lambda^{-s}) \cdot |\mathcal{S}|$ or equivalently $|\mathcal{S}| \geq (1 - c'_s\lambda^{-s}) \cdot |\mathcal{O}|$.

1.2 Computational efficiency of geometric local search

Given a parameter $\epsilon > 0$, local search with radius $\lambda = \Theta(\epsilon^{-\frac{1}{s}})$ provides a $(1+\epsilon)$-approximate solution to problems whose exchange graphs have the separator property with parameter s. This can be implemented in $n^{O(\lambda)}$ time by considering all possible local improvements, thus yielding a PTAS in time $n^{O(\epsilon^{-1/s})}$, and in particular $n^{O(\epsilon^{-2})}$ for the five problems listed on page 1.

The parameterised versions of these problems are W[1]-hard: even for unit disks, independent set is W[1]-complete [20] and dominating set is W[1]-hard [21], and dominating set of unit disks is easily reduced to our other three problems. Under the common assumption that FPT$\subsetneq$W[1], which follows from the exponential time hypothesis, these problems do not admit PTASs with time complexity $\text{poly}(n) \cdot f(\epsilon)$ for any arbitrary function f. In other words, the dependence of the exponent of n on ϵ is inevitable.

Still, this running time is prohibitively expensive, and there have been two complementary approaches towards further progress: firstly, careful implementations of local search that find local improvements more efficiently than by brute force [6]. The second, more structural approach is to better analyse the quality of solutions resulting from local search algorithms, mainly by studying the properties of exchange graphs [3].

1.3 Contributions: tightness of Theorem 1

The construction given in section 2 shows that Theorem 1 is asymptotically tight whenever $\frac{1}{s}$ is an integer.

▶ **Theorem 2.** *Given a positive integer d, there are positive constants c_d and λ_d such that, for every integer $\lambda \geq \lambda_d$, there is a family of bipartite graphs $(B_n, R_n; E_n)_{n\in\mathbb{N}}$ that*

- *are λ-expanding,*
- *have the separator property for $s = \frac{1}{d}$, and so do their subgraphs,*
- *satisfy $|B_n|, |R_n| = \Theta(n)$ and $|B_n| \geq (1 + c_d \cdot \lambda^{-\frac{1}{d}})|R_n| - o(|R_n|)$ as $n \to \infty$.*

Furthermore when $d = 2$ they are Gabriel graphs.

(A graph (V, E) is called **Gabriel** if there exists a mapping f of the vertices of V to points in the plane such that $\{v_i, v_j\} \in E$ if and only if the circumdisk of the segment $f(v_i)f(v_j)$ contains no other point of $f(V)$. Gabriel graphs are subgraphs of Delaunay triangulations and thus planar.)

▶ Remark. Since our construction for $d = 2$ is planar, previous analogues of Theorem 1 restricted to planar graphs are also tight.

1.4 Algorithmic consequences

The analysis of local search in terms of the radius that achieves a $(1+\epsilon)$-approximation is tight for the five problems listed earlier (which all had $s = \frac{1}{2}$), as well as for a few other problems with small separators (section 3).

▶ **Theorem 3** ([23]). *Local search with radius $O(\epsilon^{-2})$ is a $(1+\epsilon)$-approximation algorithm for the minimum hitting set problem for pseudodisks.*

▶ **Corollary 4.** *There is a positive constant C and a positive integer λ_0 such that for every integer $\lambda \geq \lambda_0$ there is a positive integer n_λ such that for every integer $n \geq n_\lambda$ there is a set $\mathcal{D}$ of at least n disks and two disjoint sets B and R of at least n points in $\mathbb{R}^2$ each such that both B and R are hitting sets for $\mathcal{D}$, $|B| \geq (1 + C\lambda^{-\frac{1}{2}})|R|$ and B is a λ-locally optimal solution to the hitting set problem for $\mathcal{D}$ with $\mathcal{P} = B \cup R$.*

▶ **Theorem 5** ([9]). *Local search with radius $O(\epsilon^{-2})$ is a $(1+\epsilon)$-approximation algorithm for the maximum independent set problem for pseudodisks.*

▶ **Corollary 6.** *There is a positive constant C and a positive integer λ_0 such that for every integer $\lambda \geq \lambda_0$ there is a positive integer n_λ such that for every integer $n \geq n_\lambda$ there are two independent sets B and R of at least n disks in $\mathbb{R}^2$ such that $|B| \geq (1 + C\lambda^{-\frac{1}{2}})|R|$ and R is a λ-locally optimal solution to the independent set problem in $B \cup R$.*

▶ **Theorem 7** ([5, 8]). *Local search with radius $O(\epsilon^{-2})$ is a $(1+\epsilon)$-approximation algorithm for the minimum set cover problem for disks.*

▶ **Corollary 8.** *There is a positive constant C and a positive integer λ_0 such that for every integer $\lambda \geq \lambda_0$ there is a positive integer n_λ such that for every integer $n \geq n_\lambda$ there are two independent sets B and R of at least n disks in $\mathbb{R}^2$ and a set $\mathcal{P}$ of $\Theta(|R|)$ points in $\mathbb{R}^2$ such that $|B| \geq (1 + C\lambda^{-\frac{1}{2}})|R|$ and R is a λ-locally optimal solution to the set cover problem for $\mathcal{P}$ in $B \cup R$.*

▶ **Theorem 9** ([14, 15]). *Local search with radius $O(\epsilon^{-2})$ is a $(1+\epsilon)$-approximation algorithm for the minimum dominating set problem for pseudodisks.*

▶ **Corollary 10.** *There is a positive constant C and a positive integer λ_0 such that for every integer $\lambda \geq \lambda_0$ there is a positive integer n_λ such that for every integer $n \geq n_\lambda$ there is a set $\mathcal{D}$ of disks in $\mathbb{R}^2$ and two dominating sets B and R of $\mathcal{D}$ of at least n disks each such that $|B| \geq (1 + C\lambda^{-\frac{1}{2}})|R|$ and B is a λ-locally optimal solution to the dominating set problem for $\mathcal{D}$.*

▶ **Theorem 11** ([12]). *Local search with radius $O(\epsilon^{-2})$ is a $(1+\epsilon)$-approximation algorithm for the maximum unit-capacity point-packing problem for disks.*

▶ **Corollary 12.** *There is a positive constant C and a positive integer λ_0 such that for every integer $\lambda \geq \lambda_0$ there is a positive integer n_λ such that for every integer $n \geq n_\lambda$ there are two sets B and R of at least n points in $\mathbb{R}^2$ and a set $\mathcal{D}$ of $\Theta(|R|)$ disks in $\mathbb{R}^2$ such that every disk of $\mathcal{D}$ contains one point from B and one point from R, $|B| \geq (1 + C\lambda^{-\frac{1}{2}})|R|$ and R is a λ-locally optimal solution to the unit-capacity point-packing problem for $\mathcal{D}$ in $B \cup R$.*

Borrowing a term from Arya et al. [4] we could say that: 'The locality gap for independent set of disks, dominating set of disks, etc. is $1 + \Theta(\lambda^{-1/2})$'.

2 Proof of Theorem 2

In this section we build a family of graphs that have the properties stated in Theorem 2. Namely, given parameters d, a large enough λ and n, we construct a bipartite graph G with vertex set (B, R) such that:

(a) $|R| = n + o(n)$ as $n \to +\infty$,
(b) G is λ-expanding,
(c) $|B| \geq (1 + c\lambda^{-\frac{1}{d}}) \cdot |R| - o(|R|)$ as $n \to +\infty$, where c depends only on d,
(d) any subgraph of G on m vertices has a separator of size $O(m^{1-\frac{1}{d}})$, and
(e) G is a Gabriel graph when $d = 2$.

The vertices of R are called the **red vertices**, and the vertices of B the **blue vertices**. Our construction is geometric, in that vertices correspond to points in $\mathbb{R}^d$. Thus we use the terminology vertex and point interchangeably. We denote the i-th coordinate of a point $p \in \mathbb{R}^d$ by $x_i(p)$.

Let $L \geq 2$ and t be two positive integers whose values will be fixed later as a function of the parameters d, λ and n. Let Ξ be a $L \times \cdots \times L$ regular integer grid in $\mathbb{R}^d$ consisting of the $(L+1)^d$ points in $\{0, \dots, L\}^d$. It has L^d **cells**, each defined by precisely 2^d vertices of Ξ. In every cell of Ξ, the vertex with the lexicographically minimum coordinates among the 2^d red vertices defining it is called the **anchor vertex** of that cell. Each vertex—apart from those with one of the d coordinate values equal to L—is the anchor vertex of exactly one cell, which is called its **top cell**. The cell with anchor vertex $(0, \dots, 0)$ is called the **lowest cell** of Ξ.

We define a first bipartite graph $G(d, L)$ as follows. The red vertices of $G(d, L)$ consist of the $(L+1)^d$ points of Ξ. We next place a blue vertex at the centre of each of the L^d cells of Ξ—except for the lowest cell, which contains two blue vertices with coordinates $\left(\frac{1}{4}, \dots, \frac{1}{4}, \frac{3}{4}\right)$ and $\left(\frac{3}{4}, \dots, \frac{3}{4}, \frac{1}{4}\right)$. Thus $G(d, L)$ has precisely $L^d + 1$ blue vertices. The edges of $G(d, L)$ consist of 2^d edges from each blue vertex to the 2^d red vertices of its cell. Of the two blue vertices in the lowest cell of Ξ, one is connected to all the red vertices of the cell except for $(0, \dots, 0, 1)$ (the vertex v that has $x_i(v) = 1$ if and only if $i = d$) and the other to all red vertices except for $(1, \dots, 1, 0)$.

Our second and final graph $G(d, L, t) = (B, R; E)$ is defined as a $t \times \ldots \times t$ grid composed of t^d **translates** of $G(d, L)$. Each translate of $G(d, L)$ is indexed by a vector $\vec{\tau} \in \{0, \dots, t-1\}^d$, where by $G^{\vec{\tau}}$ we denote the translate of $G(d, L)$ by $L \cdot \vec{\tau}$. The blue vertices of $G(d, L, t)$ are simply the disjoint union of the blue vertices of each $G^{\vec{\tau}}$; the red vertices are also the union of the red vertices of each $G^{\vec{\tau}}$, except that we identify duplicate red vertices shared by the boundary of two adjacent grids. See Figure 1 for an example for the case $d = 2$.

An explicit description of $G(d, L, t) = (B, R; E)$ is:

$$
\begin{aligned}
R &= \{0, \dots, tL\}^d, \\
B &= \left\{ \left(\frac{1}{2}, \dots, \frac{1}{2}\right) + \vec{x} \colon \vec{x} \in \{0, \dots, tL-1\}^d \setminus \{0, L, \dots, (t-1)L\}^d \right\} \\
&\qquad \bigcup \left\{ (\mu, \dots, \mu, 1-\mu) + L \cdot \vec{x} \colon \vec{x} \in \{0, 1, \dots, t-1\}^d,\ \mu \in \left\{\frac{1}{4}, \frac{3}{4}\right\} \right\}, \\
E &= \left\{ \{b, r\} : (b, r) \in B \times R,\ \min_{i \in \{1, \dots, d\}} |x_i(b) - x_i(r)| \leq \frac{1}{2},\ \max_{i \in \{1, \dots, d\}} |x_i(b) - x_i(r)| \leq 1 \right\}.
\end{aligned}
$$

The $L^d + 1$ blue vertices of $G^{\vec{\tau}}$ form the set $B_{\vec{\tau}}$. For the red vertices, note that the outer red vertices of each copy of $G(d, L)$ may be shared between up to 2^d translates. To avoid

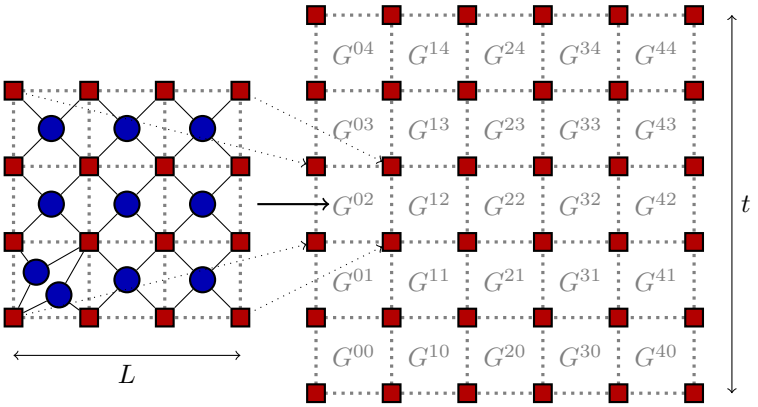


Figure 1 The graph $G(d, L)$ (shown on the left for $d = 2$ and $L = 3$) has L^d grid cells. It is the basic building block of the graph $G(d, L, t)$ (right, with $t = 5$). Square vertices are red, round vertices are blue.

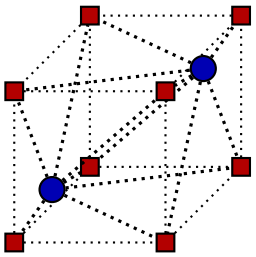


Figure 2 The three-dimensional lowest cell of $G(3, L)$.

this overlap, let $R_{\vec{\tau}}$ consist only of the L^d red vertices $v \in G^{\vec{\tau}}$ such that $x_i(v) < L(\vec{\tau}_i + 1)$ for each i. In two dimensions, this amounts to peeling off the $2L + 1$ red vertices located on the top and right boundaries of $G^{\vec{\tau}}$.

Let R_b be the set of red vertices with at least one coordinate value equal to tL. We have

$$B = \bigcup_{\vec{\tau}} B_{\vec{\tau}} \quad \text{and} \quad R = R_b \cup \bigcup_{\vec{\tau}} R_{\vec{\tau}},$$

where all unions are disjoint. Observe that

$$|B| = t^d(L^d + 1) \quad \text{and} \quad |R| = (tL + 1)^d. \tag{3}$$

Local expansion

To prove that $G(d, L, t)$ is locally expanding we fix a subset B' of B and let $R' = N(B')$ be the set of its (red) neighbours in $G(d, L, t)$. We show that $|R'| \geq |B'|$ whenever B' is smaller than some function of L and d; later we will set L such that this function turns out to be at least λ.

A grid cell is **non-empty** if it contains a vertex of B' and otherwise **empty**. A vertex of R' that belongs to R_b or whose top cell is empty is called a **boundary vertex**.

We first sketch a proof in two dimensions based on a *charging* argument (a one-to-one mapping from B' to R'): each vertex of B' is charged to a vertex of R' such that each vertex of R' receives at most one charge, implying that $|R'| \geq |B'|$. Charge each blue vertex of B' to the anchor red vertex of its cell. For those $G^{\vec{\tau}}$ containing two blue vertices in the

lowest cell, one of them remains uncharged. On the other hand, each red vertex receives one charge, except the boundary vertices which receive zero charge. Now for each $\vec{\tau}$ for which $G^{\vec{\tau}}$ contains at least two boundary red vertices charge the uncharged blue vertex in $G^{\vec{\tau}}$ (if it exists) to any one of these (at least two) boundary vertices.

There still remains an uncharged blue vertex in those $G^{\vec{\tau}}$ with less than two boundary red vertices. However, for each such $\vec{\tau}$, the number of vertices of B' in $G^{\vec{\tau}}$ must be at least $\frac{L^2}{2}$. Thus overall, there can remain at most $\frac{|B'|}{L^2/2} = \frac{2|B'|}{L^2}$ uncharged blue vertices. On the other hand, we argue that the total number of boundary red vertices is at least $c_2 \cdot \sqrt{|B'|}$, for some constant c_2. By our charging scheme, at least half of them—or $\frac{c_2}{2} \cdot \sqrt{|B'|}$—are still uncharged. Thus when $\frac{2|B'|}{L^2} \leq \frac{c_2}{2} \cdot \sqrt{|B'|}$—or equivalently $|B'| \leq c' \cdot L^4$, the number of uncharged blue vertices will be less than the number of uncharged red vertices, and we are done.

Now we present the complete proof for general d. We need two preliminary statements. Let the indicator variable $d_{\vec{\tau}}$ be 1 if both blue vertices in the lowest cell of $G^{\vec{\tau}}$ belong to B' and 0 otherwise. Also let $\delta_{\vec{\tau}}$ be the number of boundary vertices in $R_{\vec{\tau}}$. The total number of boundary vertices in R' is thus

$$\delta = |R_b \cap R'| + \sum_{\vec{\tau}} \delta_{\vec{\tau}}. \tag{4}$$

▶ **Lemma 13.** *For each index $\vec{\tau}$, if $d_{\vec{\tau}} = 1$ and $\delta_{\vec{\tau}} < 2$, then $|B' \cap B_{\vec{\tau}}| \geq \frac{L^d}{2}$.*

Proof. For such an index, B' contains both blue vertices from the lowest cell of $G^{\vec{\tau}}$ so R' contains the 2^d red vertices of this cell. If $\delta_{\vec{\tau}} = 0$, that is, $R_{\vec{\tau}}$ contains no boundary vertex, then the blue vertex in each other cell of $G^{\vec{\tau}}$ is present in B', and so B' includes all of $B_{\vec{\tau}}$, which consists of $L^d + 1$ blue vertices. It remains to consider the case when $R_{\vec{\tau}}$ contains one unique boundary vertex $v_r \in R' \cap R_{\vec{\tau}}$.

Without loss of generality, assume that $\vec{\tau} = (0, \ldots, 0)$. As both blue vertices from the lowest cell of $G^{\vec{\tau}}$ belong to B', the boundary vertex v_r cannot be the lowest vertex of $G^{\vec{\tau}}$, which has coordinates $(0, \ldots, 0)$. Thus there must be some $j \in \{1, \ldots, d\}$ for which $x_j(v_r) > 0$. Consider the grid slab Ξ' consisting of all cells whose anchor vertex v has $x_j(v) = 0$. Note that Ξ' contains the lowest cell of $G^{\vec{\tau}}$, which has two vertices of B'. Thus no other cell of Ξ' can be empty, as otherwise that would imply the existence of another boundary red vertex anchoring one of the cells of Ξ'. Now take any cell c of Ξ' whose anchor vertex differs in at least one coordinate other than x_j from v_r; there are $L^{d-1} - 1$ such cells. All the L cells of $G^{\vec{\tau}}$ whose anchor vertex only differs in the j-th coordinate value from the anchor vertex of c must also be non-empty, as otherwise it would imply the existence of a boundary red vertex in one of these L cells.

Thus there are at least $L\left(L^{d-1} - 1\right)$ non-empty cells in $G^{\vec{\tau}}$, i.e., $|B' \cap B_{\vec{\tau}}| \geq L^d - L$ which is at least $\frac{L^d}{2}$ since $L \geq 2$. ◀

Let T be the set of indices $\vec{\tau}$ with $d_{\vec{\tau}} = 1$ and $\delta_{\vec{\tau}} < 2$. As a consequence of the previous lemma, for every such $\vec{\tau} \in T$, the translate $G^{\vec{\tau}}$ contains at least $\frac{L^d}{2}$ vertices of B', and thus $|T| \leq 2|B'|L^{-d}$. Now consider the quantity $d_{\vec{\tau}} - \frac{\delta_{\vec{\tau}}}{2}$. If $\vec{\tau} \in T$, we have $d_{\vec{\tau}} = 1$ and $0 \leq \delta_{\vec{\tau}} < 2$ and so $d_{\vec{\tau}} - \frac{\delta_{\vec{\tau}}}{2}$ is at most 1. Otherwise for any $\vec{\tau} \notin T$, it is 0 or negative. Therefore

$$\sum_{\vec{\tau}} \left(d_{\vec{\tau}} - \frac{\delta_{\vec{\tau}}}{2} \right) \leq |T| \leq \frac{2|B'|}{L^d}. \tag{5}$$

An isoperimetric inequality

Consider the set S of all grid cells containing vertices of B'. As each cell contains at most two blue vertices, $|B'| \le 2|S|$. In the orthogonal projection along the i-th coordinate, S is sent to a set S_i of $(d-1)$-dimensional cells. The preimage of each cell of S_i is a column of d-dimensional cells and must contain at least one boundary vertex, so the total number δ of boundary vertices is at least $|S_i|$. The combinatorial Loomis–Whitney inequality [19] relates d- and $(d-1)$-dimensional volumes:

$$\prod_{i=1}^{d} |S_i| \ge |S|^{d-1} \ge \left(\frac{|B'|}{2}\right)^{d-1},$$

from which it follows that

$$\delta^d \ge \left(\frac{|B'|}{2}\right)^{d-1}. \tag{6}$$

Now we come to the key claim, which means that the graph $G(d, L, t)$ is $(2^{1-3d}L^{d^2})$-expanding.

▶ **Lemma 14.** *If $2^{3d-1}|B'| \le L^{d^2}$, then $|R'| \ge |B'|$.*

Proof. For every index $\vec{\tau}$, by definition, each vertex in the set $R_{\vec{\tau}} \cap R'$ either has its top cell non-empty or is a boundary vertex of $G^{\vec{\tau}}$. The number of non-empty top cells in $G^{\vec{\tau}}$ is $|B_{\vec{\tau}} \cap B'| - d_{\vec{\tau}}$, while the number of boundary vertices is $\delta_{\vec{\tau}}$. Thus

$$\begin{aligned}
|R'| &= |R_b \cap R'| + \sum_{\vec{\tau}} |R_{\vec{\tau}} \cap R'| = |R_b \cap R'| + \sum_{\vec{\tau}} \left(|B_{\vec{\tau}} \cap B'| - d_{\vec{\tau}} + \delta_{\vec{\tau}}\right) \\
&= |R_b \cap R'| + |B'| - \sum_{\vec{\tau}} (d_{\vec{\tau}} - \delta_{\vec{\tau}}) \\
&\ge |B'| - \sum_{\vec{\tau}} \left(d_{\vec{\tau}} - \frac{\delta_{\vec{\tau}}}{2}\right) + \frac{1}{2}\left(|R_b \cap R'| + \sum_{\vec{\tau}} \delta_{\vec{\tau}}\right) \\
&= |B'| - \sum_{\vec{\tau}} \left(d_{\vec{\tau}} - \frac{\delta_{\vec{\tau}}}{2}\right) + \frac{\delta}{2}.
\end{aligned}$$

Use the lower bounds (5) and (6) for the second and third summands:

$$|R'| \ge |B'| - \frac{2|B'|}{L^d} + \frac{1}{2}\left(\frac{|B'|}{2}\right)^{(d-1)/d},$$

and the result $|R'| \ge |B'|$ follows when

$$\begin{aligned}
\frac{2|B'|}{L^d} &\le \frac{1}{2}\left(\frac{|B'|}{2}\right)^{(d-1)/d} \\
2^{3-1/d}|B'|^{1/d} &\le L^d
\end{aligned}$$

or equivalently $2^{3d-1}|B'| \le L^{d^2}$. ◀

Ball graph structure

A **ball graph** is the intersection graph of a family of n balls in $\mathbb{R}^d$ and is p-ply if it has no clique of size $p+1$. Such graphs have separators of size $O(p^{1/d}n^{1-\frac{1}{d}})$ [22].

A bounded-ply[5] ball graph is obtained from $G(d,L,t)$ by only adding some edges: put a d-dimensional ball of radius $\frac{\sqrt{d}}{4}$ at each vertex of $G(d,L,t)$. The resulting edge set includes that of $G(d,L,t)$—they coincide when $d \leq 3$—so that $G(d,L,t)$ inherits separator properties of ball graphs. In other words, any subgraph of $G(d,L,t)$ on m vertices has a separator of size $O(m^{1-\frac{1}{d}})$ (this is property (d)).

Gabriel graph structure

For $d=2$, the circumdisk of each blue–red edge in $G(d,L,t)$ contains no vertex but its endpoints, so $G(d,L,t)$ is a Gabriel graph and property (e) is proved.

▶ Remark. With the understanding that a one-dimensional cell is an interval, the construction covers the case $d=1$. The graph $G(1,L,t)$ is a path of length $2tL+1$, seen as blue–red bipartite, with every L-th blue vertex duplicated. It has $|R| = tL+1$ and $|B| = t(L+1)$ and is $(L+2)$-expanding.

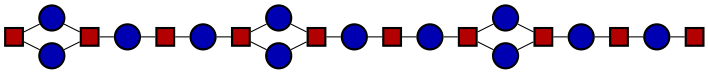


Figure 3 The graph $G(1,3,3)$.

Setting parameters and concluding the proof

Given d, λ and n, choose

$$L = \max\left\{2, \left\lceil \left(2^{3d-1}\lambda\right)^{1/d^2} \right\rceil\right\} \text{ and } t = \lceil n^{1/d} L^{-1} \rceil.$$

Note that L does not depend on n and L^d is $\Theta(\lambda^{\frac{1}{d}})$ when $\lambda \to +\infty$. Using (3), we obtain (a) and (c):

$$\begin{aligned} |R| &= (tL+1)^d = n + o(n), \\ \frac{|B|}{|R|} &= \frac{t^d\left(L^d+1\right)}{(tL+1)^d} = 1 + \frac{1}{L^d} + o(1) \text{ as } n \to +\infty \\ &\geq 1 + c_d\lambda^{-\frac{1}{d}} + o_n(1) \text{ for } \lambda \geq \lambda_d \end{aligned}$$

where the positive constants c_d and λ_d depend only on d. Since $2^{1-3d}L^{d^2} \geq \lambda$, it follows from Lemma 14 that (b) holds: G is at least λ-expanding. This achieves the proof of Theorem 2.

3 Algorithmic consequences

3.1 Geometric problems in the plane

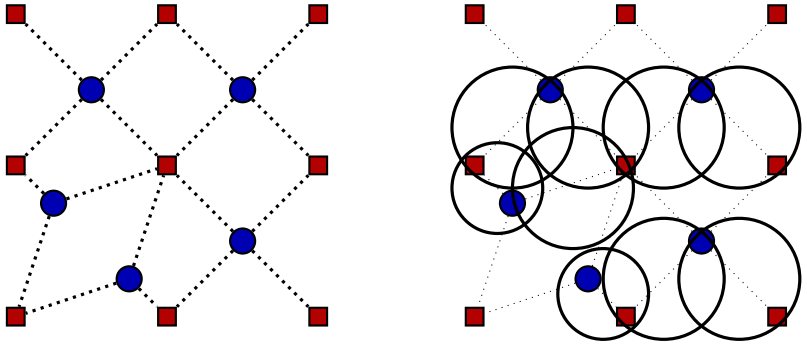
We construct arbitrarily large instances of our five optimisation problems with a λ-locally optimal solution that is $1+\Omega(\lambda^{-1/2})$ times worse than the optimal solution. Since our

[5] Bounded by a function of d only: the largest number of vertices of $B \cup R$ included in a same ball of radius $\frac{\sqrt{d}}{2}$. See e.g. [11] for estimates on such bounds.

instances consist of proper disks rather than just families of pseudodisks, the bound applies also to the restrictions of these problems to disk families.

For $d = 2$ and any given $\lambda \geq \lambda_d$ and n, let $G = (B, R; E)$ be the planar and λ-expanding graph $(B_n, R_n; E_n)$ described in Theorem 2 and built in section 2. Our instances are based on G: its vertex sets are associated with feasible solutions of the problems. It then suffices to check that the solution associated with B_n (for minimisation problems) or R_n (for maximisation problems) is locally optimal.

3.1.1 Hitting set for pseudodisks



(a) Detail of the graph G used in 'bad' instances.

(b) Hitting set (drawing only a few disks for readability).

Figure 4 A 'tight' instance for the hitting set problem.

▶ **Theorem 3** ([23]). *Local search with radius $O(\epsilon^{-2})$ is a $(1+\epsilon)$-approximation algorithm for the minimum hitting set problem for pseudodisks.*

▶ **Corollary 4.** *There is a positive constant C and a positive integer λ_0 such that for every integer $\lambda \geq \lambda_0$ there is a positive integer n_λ such that for every integer $n \geq n_\lambda$ there is a set $\mathcal{D}$ of at least n disks and two disjoint sets B and R of at least n points in $\mathbb{R}^2$ each such that both B and R are hitting sets for $\mathcal{D}$, $|B| \geq (1 + C\lambda^{-\frac{1}{2}})|R|$ and B is a λ-locally optimal solution to the hitting set problem for $\mathcal{D}$ with $\mathcal{P} = B \cup R$.*

Proof. Recall that the circumdisk of each edge of G contains only its two endpoints. The input consists of all such disks, with $\mathcal{P} = B \cup R$, so that the hitting sets are exactly the vertex covers of G. By construction both B and R are feasible solutions.

On this instance, a λ-local improvement for B would remove a set B' of blue vertices with $|B'| \leq \lambda$. To preserve the hitting set property, it would then need to add to the solution the red endpoints of all edges with their blue endpoint in B', i.e. the set $N(B')$. Because the graph is λ-expanding, there are at least $|B'|$ such red neighbours: B is λ-locally optimal. ◀

3.1.2 Independent set of pseudodisks

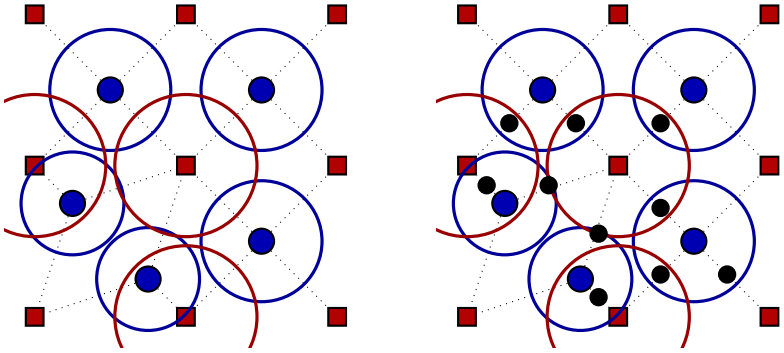
▶ **Theorem 5** ([9]). *Local search with radius $O(\epsilon^{-2})$ is a $(1+\epsilon)$-approximation algorithm for the maximum independent set problem for pseudodisks.*

▶ **Corollary 6.** *There is a positive constant C and a positive integer λ_0 such that for every integer $\lambda \geq \lambda_0$ there is a positive integer n_λ such that for every integer $n \geq n_\lambda$ there are two*

independent sets B and R of at least n disks in $\mathbb{R}^2$ such that $|B| \geq (1 + C\lambda^{-\frac{1}{2}})|R|$ and R is a λ-locally optimal solution to the independent set problem in $B \cup R$.

Proof. Realise the graph G as an intersection graph of red and blue disks. (Because it is planar, the disks could even be taken interior-disjoint by the Koebe–Andreev–Thurston theorem.) The independent sets of disks correspond to the independent sets of G. Since G is bipartite both the blue and red families of disks form independent sets, and the red solution is $(\lambda - 1)$-locally optimal—in maximisation terms: a $(\lambda - 1)$-local improvement for the red solution would remove a set R' of up to $\lambda - 1$ red disks and replace them with a set B' of blue disks such that $N(B') \subseteq R'$ (to preserve independence) and $|B'| > |R'|$. If there exists a subset $B'' \subseteq B'$ of size $|R'| + 1$, which is at most λ, then since G is λ-expanding such a set has $|B''| \leq |N(B'')| \leq |R'|$, a contradiction. Thus R is a $(\lambda - 1)$-locally optimal solution. ◀

3.1.3 Set cover for disks



(a) Independent set. **(b)** Set cover.

Figure 5 'Tight' instances for independent set and set cover with disks.

▶ **Theorem 7** ([5, 8]). *Local search with radius $O(\epsilon^{-2})$ is a $(1 + \epsilon)$-approximation algorithm for the minimum set cover problem for disks.*

▶ **Corollary 8.** *There is a positive constant C and a positive integer λ_0 such that for every integer $\lambda \geq \lambda_0$ there is a positive integer n_λ such that for every integer $n \geq n_\lambda$ there are two independent sets B and R of at least n disks in $\mathbb{R}^2$ and a set $\mathcal{P}$ of $\Theta(|R|)$ points in $\mathbb{R}^2$ such that $|B| \geq (1 + C\lambda^{-\frac{1}{2}})|R|$ and R is a λ-locally optimal solution to the set cover problem for $\mathcal{P}$ in $B \cup R$.*

Proof. As in the proof of Corollary 6, realise G as an intersection graph of blue and red disks. Take for $\mathcal{P}$ one point from each blue–red intersection. The set covers for this instance are exactly the vertex covers of G. ◀

3.1.4 Dominating set of pseudodisks

▶ **Theorem 9** ([14, 15]). *Local search with radius $O(\epsilon^{-2})$ is a $(1+\epsilon)$-approximation algorithm for the minimum dominating set problem for pseudodisks.*

▶ **Corollary 10.** *There is a positive constant C and a positive integer λ_0 such that for every integer $\lambda \geq \lambda_0$ there is a positive integer n_λ such that for every integer $n \geq n_\lambda$ there is a set $\mathcal{D}$ of disks in $\mathbb{R}^2$ and two dominating sets B and R of $\mathcal{D}$ of at least n disks each such that*

$|B| \geq (1 + C\lambda^{-\frac{1}{2}})|R|$ and B is a λ-locally optimal solution to the dominating set problem for $\mathcal{D}$.

Proof. The instance that was proposed for set cover (Figure 5b) becomes an instance of dominating set when the points of $\mathcal{P}$ are seen as zero-radius disks, i.e. take $\mathcal{D} = \mathcal{P} \cup B \cup R$. A feasible solution that involves some of the zero-radius disks of $\mathcal{P}$ can be transformed into a solution of at most the same cardinality whose support is entirely blue and red since the disks of $\mathcal{P}$ are fully included in the other disks. Thus it suffices to examine the efficiency of local search on blue–red solutions. The blue–red dominating sets of this instance are exactly the covers of points by blue and red disks. ◀

3.1.5 Unit-capacity packing problems

▶ **Theorem 11** ([12]). *Local search with radius $O(\epsilon^{-2})$ is a $(1 + \epsilon)$-approximation algorithm for the maximum unit-capacity point-packing problem for disks.*

▶ **Corollary 12.** *There is a positive constant C and a positive integer λ_0 such that for every integer $\lambda \geq \lambda_0$ there is a positive integer n_λ such that for every integer $n \geq n_\lambda$ there are two sets B and R of at least n points in $\mathbb{R}^2$ and a set $\mathcal{D}$ of $\Theta(|R|)$ disks in $\mathbb{R}^2$ such that every disk of $\mathcal{D}$ contains one point from B and one point from R, $|B| \geq (1 + C\lambda^{-\frac{1}{2}})|R|$ and R is a λ-locally optimal solution to the unit-capacity point-packing problem for $\mathcal{D}$ in $B \cup R$.*

Proof. Take for $\mathcal{D}$ the set of all disks associated with the edges, as in the hitting set instance (see Figure 4b). Since every such disk contains only two points of $\mathcal{P}$, the 'unit-capacity point-packings' of this instance are exactly the independent sets of G. The result then follows from the analysis in Corollary 6. ◀

The dual problem, defined in [12], is the unit-capacity disk-packing problem, where we are given a set $\mathcal{D}$ of disks and a set $\mathcal{P}$ of points and we must return a largest subset of $\mathcal{D}$ that covers every point of $\mathcal{P}$ at most once.

▶ **Corollary 15.** *There is a positive constant C and a positive integer λ_0 such that for every integer $\lambda \geq \lambda_0$ there is a positive integer n_λ such that for every integer $n \geq n_\lambda$ there are two sets B and R of at least n disks each and a set $\mathcal{P}$ of $\Theta(|R|)$ points such that every point of $\mathcal{P}$ is contained in one disk from B and one disk from R, $|B| \geq (1 + C\lambda^{-\frac{1}{2}})|R|$ and R is a λ-locally maximal solution to the unit-capacity disk-packing problem for $B \cup R$ in $\mathcal{P}$.*

3.2 Problems with hereditary separators

The paper by Har-Peled and Quanrud [16] is to the best of our knowledge the most extensive study of geometric local search in non-planar settings. The authors study graphs with polynomial expansion[6], which have strongly sub-linear separators, and in particular intersection graphs of low-density families of objects[7].

We are still able to give some lower bounds on the local search radii that achieve PTASs. Fix positive integers d, $\lambda \geq \lambda_d$ and n, and let G be the λ-expanding graph built in section 2 on vertex sets B_n and R_n that has $|B_n|, |R_n| = \Theta(n)$ and achieves $|B_n| \geq (1 + c\lambda^{-1/d} - o(1))|R_n|$. Recall that G and its subgraphs have the separator property with $s = 1/d$.

By combining Theorem 3.2.1 and Lemma 2.2.9 from [16], we obtain the following.

[6] A survey on expansion and sparsity is the book by Nešetřil and Ossona de Mendez [24].

[7] A family of objects in $\mathbb{R}^d$ has **density** ρ if for any $r \geq 0$ any ball of diameter r intersects at most ρ objects of diameter larger than r and **depth** D if no point of $\mathbb{R}^d$ is contained in $D + 1$ objects.

▶ **Theorem 16.** *On graphs with hereditary separators of size $O(n^{1-s})$, local search with radius $O(\epsilon^{-s})$ is a $(1+\epsilon)$-approximation algorithm for maximum independent set.*

▶ **Corollary 17.** *For every positive integers d and λ, there are arbitrarily large bipartite graphs on vertex sets (B,R) with hereditary separators of size $O(n^{1-1/d})$ such that $|B| \geq 1+\Omega(\lambda^{-1/d})|R|$ and R is a λ-locally maximal independent set.*

Proof. Since the graph G is bipartite, both B_n and R_n are independent sets, and by the same analysis as in the proof of Corollary 6 the feasible solution R_n is $(\lambda-1)$-locally optimal. ◀

▶ **Theorem 18** ([16])**.** *On graphs with hereditary separators of size $O(n^{1-s})$, local search with radius $O(\epsilon^{-O(1)})$ is a $(1+\epsilon)$-approximation algorithm for minimum vertex cover.*

▶ **Corollary 19.** *For every positive integers d and λ, there are arbitrarily large bipartite graphs on vertex sets (B,R) with hereditary separators of size $O(n^{1-1/d})$ such that $|B| \geq 1+\Omega(\lambda^{-1/d})|R|$ and B is a λ-locally minimal vertex cover.*

Proof. In G both B_n and R_n are vertex covers. Since G is λ-expanding, B_n is λ-locally optimal. ◀

3.3 Matchings and local versions of Hall's theorem

With our terminology, Hall's theorem is as follows.

▶ **Theorem 20** (Hall's marriage theorem)**.** *Any bipartite graph on vertex sets (B,R) that is $|B|$-expanding has a matching with $|B|$ edges.*

Restricting the condition to λ-expansion for some fixed λ breaks this property—the matchings of $K_{|B|,\lambda}$ have at most λ edges. However it was observed by Antunes, Mathieu and Mustafa [3] that a strengthening of Hall's theorem holds for planar graphs.

▶ **Theorem 21.** *There is an absolute constant $c>0$ such that, for every given integer $\lambda \geq 3$, any bipartite* planar *graph on vertex sets (B,R) that is λ-expanding has a matching with at least $(1-c\lambda^{-\frac{1}{2}})|B|$ edges.*

Now it follows from our constructions that this is tight.

▶ **Corollary 22.** *There are absolute constants $c_0, \lambda_0 > 0$ such that, for every given integer $\lambda \geq \lambda_0$, some bipartite, λ-expanding planar graph on vertex sets (B,R) does not have matchings with more than $(1-c_0\lambda^{-\frac{1}{2}})|B|$ edges.*

4 Perspectives and open questions

We emphasise that our results apply to standard, non-specialised local-search techniques. Although the approximation quality of a previously successful one-size-fits-all approach cannot be improved, custom algorithms tailored for specific problems can bypass this bound, especially when the exchange graphs are extremely sparse. For example we do not know whether our constructions can be transformed into a local-search-defeating instance for the problem of terrain guarding, a question that can be formulated as follows.

▶ Question. Are the exchange graphs of Gibson et al. [13] for terrain guarding sparser than other planar graphs? What is the minimum size of their separators?

References

1 Pankaj K. Agarwal and Nabil H. Mustafa. Independent set of intersection graphs of convex objects in 2D. *Computational Geometry*, 34(2):83–95, 2006. doi:10.1016/j.comgeo.2005.12.001.

2 Noga Alon, Paul Seymour, and Robin Thomas. A separator theorem for nonplanar graphs. *Journal of the American Mathematical Society*, 3(4), 10 1990. doi:10.1090/S0894-0347-1990-1065053-0.

3 Daniel Antunes, Claire Mathieu, and Nabil H. Mustafa. Combinatorics of local search: An optimal 4-local hall's theorem for planar graphs. In Kirk Pruhs and Christian Sohler, editors, *25th Annual European Symposium on Algorithms (ESA 2017)*, volume 87 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 8:1–8:13, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ESA.2017.8.

4 Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k-median and facility location problems. *SIAM Journal on Computing*, 33(3):544–562, 2004. doi:10.1137/S0097539702416402.

5 Rom Aschner, Matthew J. Katz, Gila Morgenstern, and Yelena Yuditsky. Approximation schemes for covering and packing. In Subir Kumar Ghosh and Takeshi Tokuyama, editors, *WALCOM: Algorithms and Computation: 7th International Workshop, WALCOM 2013, Kharagpur, India, February 14-16, 2013. Proceedings*, pages 89–100, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. doi:10.1007/978-3-642-36065-7_10.

6 Norbert Bus, Shashwat Garg, Nabil H. Mustafa, and Saurabh Ray. Limits of local search: Quality and efficiency. *Discrete & Computational Geometry*, 57(3):607–624, 4 2017. doi:10.1007/s00454-016-9819-x.

7 Sergio Cabello and David Gajser. Simple PTAS's for families of graphs excluding a minor. *Discrete Applied Mathematics*, 189:41–48, 2015. doi:10.1016/j.dam.2015.03.004.

8 Timothy M. Chan and Elyot Grant. Exact algorithms and APX-hardness results for geometric packing and covering problems. *Comput. Geom.*, 47(2, Part A):112–124, 2014. Special Issue: 23rd Canadian Conference on Computational Geometry (CCCG11). doi:10.1016/j.comgeo.2012.04.001.

9 Timothy M. Chan and Sariel Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. *Discrete & Computational Geometry*, 48(2):373–392, 09 2012. doi:10.1007/s00454-012-9417-5.

10 Brent N. Clark, Charles J. Colbourn, and David S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86(1):165–177, 1990. doi:10.1016/0012-365X(90)90358-O.

11 John Conway and Neil J. A. Sloane. *Sphere Packings, Lattices and Groups*. Grundlehren der mathematischen Wissenschaften. Springer-Verlag New York, New York, NY, USA, 1999. doi:10.1007/978-1-4757-6568-7.

12 Alina Ene, Sariel Har-Peled, and Benjamin Raichel. Geometric packing under non-uniform constraints. In *Symposium on Computational Geometry 2012, SoCG '12, Chapel Hill, NC, USA, June 17-20, 2012*, pages 11–20, 2012. doi:10.1145/2261250.2261253.

13 Matt Gibson, Gaurav Kanade, Erik Krohn, and Kasturi Varadarajan. An approximation scheme for terrain guarding. In Irit Dinur, Klaus Jansen, Joseph Naor, and José Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques: 12th International Workshop, APPROX 2009, and 13th International Workshop, RANDOM 2009, Berkeley, CA, USA, August 21-23, 2009. Proceedings*, pages 140–148, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. doi:10.1007/978-3-642-03685-9_11.

14 Matt Gibson and Imran A. Pirwani. Algorithms for dominating set in disk graphs: Breaking the $\log n$ barrier. In Mark de Berg and Ulrich Meyer, editors, *Proceedings of the 18th*

Annual European Symposium on Algorithms (ESA), pages 243–254, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. doi:10.1007/978-3-642-15775-2_21.

15 Sathish Govindarajan, Rajiv Raman, Saurabh Ray, and Aniket Basu Roy. Packing and covering with non-piercing regions. In Piotr Sankowski and Christos Zaroliagis, editors, *Proceedings of the 22nd Annual European Symposium on Algorithms (ESA)*, volume 57 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 47:1–47:17, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ESA.2016.47.

16 Sariel Har-Peled and Kent Quanrud. Approximation algorithms for polynomial-expansion and low-density graphs. In *Proceedings of the 23rd Annual European Symposium on Algorithms (ESA)*, pages 717–728, 2015. doi:10.1007/978-3-662-48350-3_60.

17 Sariel Har-Peled and Kent Quanrud. Notes on approximation algorithms for polynomial-expansion and low-density graphs, 2016. arXiv:1603.03098.

18 Dorit S. Hochbaum and Wolfgang Maass. Fast approximation algorithms for a nonconvex covering problem. *Journal of Algorithms*, 8(3):305–323, 1987. doi:10.1016/0196-6774(87)90012-5.

19 Lynn H. Loomis and Hassler Whitney. An inequality related to the isoperimetric inequality. *Bulletin of the American Mathematical Society*, 55(10):961–962, 1949. doi:10.1090/S0002-9904-1949-09320-5.

20 Dániel Marx. Efficient approximation schemes for geometric problems? In Gerth Stølting Brodal and Stefano Leonardi, editors, *Algorithms – ESA 2005: 13th Annual European Symposium, Palma de Mallorca, Spain, October 3-6, 2005. Proceedings*, pages 448–459, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. doi:10.1007/11561071_41.

21 Dániel Marx. Parameterized complexity of independence and domination on geometric graphs. In Hans L. Bodlaender and Michael A. Langston, editors, *Parameterized and Exact Computation: Second International Workshop, IWPEC 2006, Zürich, Switzerland, September 13-15, 2006. Proceedings*, pages 154–165, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. doi:10.1007/11847250_14.

22 Gary L. Miller, Shang-Hua Teng, William Thurston, and Stephen A. Vavasis. Separators for sphere-packings and nearest neighbor graphs. *J. ACM*, 44(1):1–29, 1997. doi:10.1145/256292.256294.

23 Nabil H. Mustafa and Saurabh Ray. Improved results on geometric hitting set problems. *Discrete & Computational Geometry*, 44(4):883–895, 12 2010. doi:10.1007/s00454-010-9285-9.

24 Jaroslav Nešetřil and Patrice Ossona de Mendez. *Sparsity*, volume 28 of *Algorithms and Combinatorics*. Springer-Verlag Berlin Heidelberg, 2012. doi:10.1007/978-3-642-27875-4.

Odd Yao-Yao Graphs are Not Spanners

Yifei Jin
Tsinghua University
Beijing 100084, China
jin-yf13@mails.tsinghua.edu.cn

Jian Li
Tsinghua University
Beijing 100084, China
corresponding author: lijian83@mail.tsinghua.edu.cn

Wei Zhan
Princeton University
Princeton, NJ 08544, USA
weizhan@cs.princeton.edu

Abstract

It is a long standing open problem whether Yao-Yao graphs YY_k are all spanners [Li et al. 2002]. Bauer and Damian [Bauer and Damian, 2012] showed that all YY_{6k} for $k \geq 6$ are spanners. Li and Zhan [Li and Zhan, 2016] generalized their result and proved that all even Yao-Yao graphs YY_{2k} are spanners (for $k \geq 42$). However, their technique cannot be extended to odd Yao-Yao graphs, and whether they are spanners are still elusive. In this paper, we show that, surprisingly, for any integer $k \geq 1$, there exist odd Yao-Yao graph YY_{2k+1} instances, which are not spanners.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Odd Yao-Yao Graph, Spanner, Counterexample

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.49

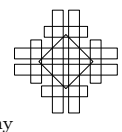
Related Version A full version of this paper is available at `https://arxiv.org/abs/1704.03132`.

Funding This research is supported in part by the National Basic Research Program of China Grant 2015CB358700, the National Natural Science Foundation of China Grant 61772297, 61632016, 61761146003, and a grant from Microsoft Research Asia.

1 Introduction

Let $\mathcal{P}$ be a set of points in the Euclidean plane $\mathbb{R}^2$. The complete Euclidean graph defined on set $\mathcal{P}$ is the edge-weighted graph with vertex set $\mathcal{P}$ and edges connecting all pairs of points in $\mathcal{P}$, where the weight of each edge is the Euclidean distance between its two end points. Storing the complete graph requires quadratic space, which is very expensive. Hence, it is desirable to use a sparse subgraph to approximate the complete graph. This is a classical and well-studied topic in computational geometry (see e.g., [1, 19, 26, 32, 34]). In this paper, we study the so called *geometric t-spanner*, formally defined as follows (see e.g., [29]).

▶ **Definition 1** (Geometric t-Spanner)**.** A graph G is a *geometric t-spanner* of the complete Euclidean graph, if (1) G is a subgraph of the complete Euclidean graph; and (2) for any pair of points p and q in $\mathcal{P}$, the shortest path between p and q in G is no longer than t times the Euclidean distance between p and q.




The factor t is called the *stretch factor* or *dilation factor* of the spanner in the literature. If the maximum degree of G is bounded by a constant k, we say that G is a *bounded-degree spanner*. The concept of geometric spanners was first proposed by L.P. Chew [10]. See the comprehensive survey by Eppstein [16] for related topics about geometric spanners. Geometric spanners have found numerous applications in wireless ad hoc and sensor networks. We refer the readers to the books by Li [24] and Narasimhan and Smid [27] for more details.

Yao graphs are one of the first approximations of complete Euclidean graphs, introduced independently by Flinchbaugh and Jones [18] and Yao [34].

▶ **Definition 2** (Yao Graph Y_k)**.** Let k be a fixed integer. Given a set of points $\mathcal{P}$ in the Euclidean plane $\mathbb{R}^2$, the Yao graph $\mathsf{Y}_k(\mathcal{P})$ is defined as follows. Let $C_u(\gamma_1, \gamma_2]$ be the cone with apex u, which consists of the rays with polar angles in the half-open interval $(\gamma_1, \gamma_2]$. For each $u \in \mathcal{P}$, $\mathsf{Y}_k(\mathcal{P})$ contains an edge connecting u to a nearest neighbor v in each cone $C_u(j\theta, (j+1)\theta]$, for $\theta = 2\pi/k$ and $j \in [0, k-1]$. We generally consider Yao graphs as undirected graphs. For a *directed Yao graph*, we add directed edge $\overrightarrow{uv}$ to the graph instead.

Molla [15] showed that Y_2 and Y_3 may not be spanners. On the other hand, it has been proven that all Y_k for $k \geq 4$ are spanners. Bose et al. [6] proved that Y_4 is a 663-spanner. Damian and Nelavalli [13] improved this to 54.6 recently. Barba et al. [2] showed that Y_5 is a 3.74-spanner. Damian and Raudonis [14] proved that the Y_6 graph is a 17.64 spanner. Li et al. [25, 26] first proved that all $\mathsf{Y}_k, k > 6$ are spanners with stretch factor at most $1/(1 - 2\sin(\pi/k))$. Later Bose et al. [6, 7] also obtained the same result independently. Recently, Barba et al. [2] reduced the stretch factor of Y_6 from 17.6 to 5.8 and improved the stretch factors to $1/(1 - 2\sin(3\pi/4k))$ for odd $k \geq 7$.

However, a Yao graph may not have bounded degree. This can be a serious limitation in certain wireless network applications since each node has very limited energy and communication capacity, and can only communicate with a small number of neighbors. To address the issue, Li et al. [26] introduced *Yao-Yao* graphs (or *Sparse-Yao* graphs in the literature). A Yao-Yao graph $\mathsf{YY}_k(\mathcal{P})$ is obtained by removing some edges from $\mathsf{Y}_k(\mathcal{P})$ as follows:

▶ **Definition 3** (Yao-Yao Graph YY_k)**.** (1) Construct the directed Yao graph, as in Definition 2. (2) For each node u and each cone rooted at u containing two or more incoming edges, retain a shortest incoming edge and discard the other incoming edges in the cone. We can see that the maximum degree in $\mathsf{YY}_k(\mathcal{P})$ is upper-bounded by $2k$.

As opposed to Yao graphs, the spanning property of Yao-Yao graphs is not well understood yet. Li et al. [26] provided some empirical evidence, suggesting that YY_k graphs are t-spanners for some sufficiently large constant k. However, there is no theoretical proof yet, and it is still an open problem [4, 23, 24, 26]. It is also listed as Problem 70 in the Open Problems Project.[1]

▶ **Conjecture 4** (see [4])**.** There exists a constant k_0 such that for any integer $k > k_0$, any Yao-Yao graph YY_k is a geometric spanner.

Now, we briefly review the previous results about Yao-Yao graphs. It is known that YY_2 and YY_3 may not be spanners since Y_2 and Y_3 may not be spanners [15]. Damian and Molla [12, 15] proved that $\mathsf{YY}_4, \mathsf{YY}_6$ may not be spanners. Bauer et al. [2] proved that YY_5 may not be spanners. On the positive side, Bauer and Damian [4] showed that for any integer

[1] `http://cs.smith.edu/~orourke/TOPP/P70.html`

$k \geq 6$, any Yao-Yao graph YY_{6k} is a spanner with the stretch factor at most 11.67 and the factor becomes 4.75 for $k \geq 8$. Recently, Li and Zhan [23] proved that for any integer $k \geq 42$, any even Yao-Yao graph YY_{2k} is a spanner with the stretch factor $6.03 + O(k^{-1})$.

From these positive results, it is quite tempting to believe Conjecture 4. However, we show in this paper that, surprisingly, Conjecture 4 is false for odd Yao-Yao graphs.

▶ **Theorem 5.** *For any $k \geq 1$, there exists a class of point set instances $\{\mathcal{P}_m\}_{m\in\mathbb{Z}^+}$ such that the stretch factor of $\mathsf{YY}_{2k+1}(\mathcal{P}_m)$ cannot be bounded by any constant, as m approaches infinity.*[2]

Related work. It has been proven that in some special cases, Yao-Yao graphs are spanners [11,21,22,33]. Specifically, it was shown that YY_k graphs are spanners in *civilized graphs*, where the ratio of the maximum edge length to the minimum edge length is bounded by a constant [21,22].

Besides the Yao and Yao-Yao graph, the Θ-*graph* is another common geometric t-spanner. The difference between Θ-graphs and Yao graphs is that in a Θ-graph, the nearest neighbor to u in a cone C is a point $v \neq u$ lying in C and minimizing the Euclidean distance between u and the orthogonal projection of v onto the bisector of C. It is known that except for Θ_2 and Θ_3 [15], for $k = 4$ [3], 5 [8], 6 [5], ≥ 7 [9,28], Θ_k-graphs are all geometric spanners. We note that, unfortunately, the degrees of Θ-graphs may not be bounded.

Recently, some variants of geometric t-spanners such as weak t-spanners and power t-spanners have been studied. In weak t-spanners, the path between two points may be arbitrarily long, but must remain within a disk of radius t-times the Euclidean distance between the points. It is known that all Yao-Yao graphs YY_k for $k > 6$ are weak t-spanners [20,30,31]. In power t-spanners, the Euclidean distance $|\cdot|$ is replaced by $|\cdot|^\kappa$ with a constant $\kappa \geq 2$. Schindelhauer et al. [30,31] proved that for $k > 6$, all Yao-Yao graphs YY_k are power t-spanners for some constant t. Moreover, it is known that any t-spanner is also a weak t_1-spanner and a power t_2-spanner for some t_1, t_2 depending only on t. However, the converse is not true [31].

Our counterexample is inspired by the concept of fractals. Fractals have been used to construct examples for β-skeleton graphs with unbounded stretch factors [17]. Here a β-skeleton graph is defined to contain exactly those edges ab such that no point c forms an angle $\angle acb$ greater than $\sin^{-1} 1/\beta$ if $\beta > 1$ or $\pi - \sin^{-1} \beta$ if $\beta < 1$. Schindelhauer et al. [31] used the same example to prove that there exist graphs which are weak spanners but not t-spanners. However, their examples cannot serve as counterexamples to the conjecture that odd Yao-Yao graphs are spanners.

2 Overview of our counterexample construction

We first note that both the counterexamples for YY_3 and YY_5 are not weak t-spanners [2,15]. However, Yao-Yao graphs YY_k for $k \geq 7$ are all weak t-spanners [20, 30, 31]. Hence, to construct the counterexamples for YY_k for $k \geq 7$, the previous ideas for YY_3 and YY_5 cannot be used. We will construct a class of instances $\{\mathcal{P}_m\}_{m\in\mathbb{Z}^+}$ such that all points in $\mathcal{P}_m$ are placed in a bounded area. Meanwhile, there exist shortest paths in $\mathsf{YY}_{2k+1}(\mathcal{P}_m)$ whose lengths approach infinity as m approaches infinity.

[2] Here, m is a parameter in our recursive construction. We will explain it in detail in Section 3. Roughly speaking, m is the level of recursion, and the number of points in $\mathcal{P}_m$ increases with m.

Our example contains two types of points, called *normal points* and *auxiliary points*. Denote them by $\mathcal{P}_m^{\mathrm{n}}$ and $\mathcal{P}_m^{\mathrm{a}}$ respectively and $\mathcal{P}_m = \mathcal{P}_m^{\mathrm{n}} \cup \mathcal{P}_m^{\mathrm{a}}$. The normal points form the basic skeleton, and the auxiliary points are used to break the edges connecting any two normal points that are far apart.

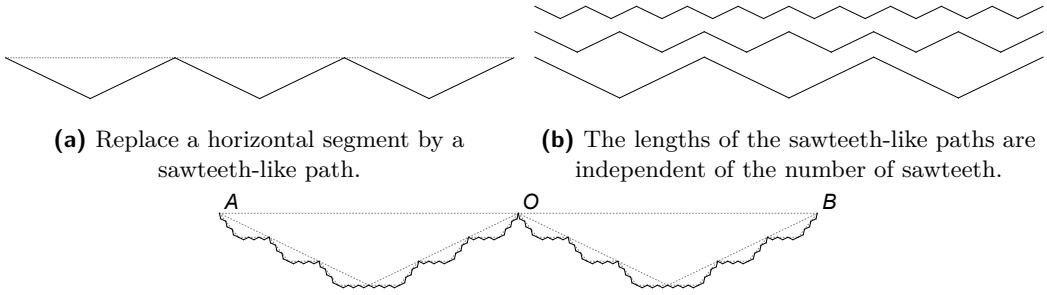
We are inspired by the concept of fractals to construct the normal points. A fractal can be contained in a bounded area, but its length may diverge. In our counterexample, the shortest path between two specific normal points is a fractal-like polygonal path. Here a polygonal path refers to a curve specified by a sequence of points and consists of the line segments connecting the consecutive points. Suppose the two specific points are A and B, AB is horizontal, and $|AB| = 1$. When $m = 0$, the polygonal path is just the line segment AB. When m increases by one, we replace each line segment in the current polygonal path by a *sawteeth-like* path (see Figure 1a). If the angle between each segment of the sawteeth-like path and the base segment (i.e., the one which is replaced) is γ, the total length of the path increases by a factor of $\cos^{-1}\gamma$. An important observation here is that the factor is independent of the number of sawteeth (see Figure 1b). If we continue this process directly, the length of the resulting path would increase to infinity as m approaches infinity since $\cos^{-1}\gamma > 1$ (see Figure 1c). However, we need to make sure that such a path is indeed in a Yao-Yao graph and it is indeed the shortest path from A to B. There are two technical difficulties we need to overcome.

1. As m increases, the polygonal path may intersect itself. See Figure 1d. The polygonal path intersects itself around the point O. This is relatively easy to handle: we do not recurse for those segments that may cause self-intersection. See Figure 1e. We do not replace the bold segment further. We need to make sure that the total length of such segments is proportionally small (so that the total length can keep increasing as m increases).
2. In the Yao-Yao graph defined over the normal points constructed in the recursion, there may be some edges connecting points that are far apart. Actually, how to break such edges is the main difficulty of the problem. We outline the main techniques below.

First, we *do not* replace all current segments using the same sawteeth, like in the usual fractal construction. Actually, for each segment, we will choose a polygonal path such that the paths have different numbers of sawteeth and the sizes of the sawteeth in the path may not be the same. See Figure 1f. Finally, we construct them in a specific sequential order. Actually, we organize the normal points in an m-level *recursion tree* $\mathcal{T}$ and generate them in a DFS preorder traversal of the tree. We describe the details in Section 3.

Second, we group the normal points into a collection of sets such that each normal point belongs to exactly one set. We call such a set a *hinge set*. Refer to Figure 6 for an overview. Then, we specify a *total order* of the hinge sets. Call the edges in the Yao-Yao graph $\mathsf{YY}_{2k+1}(\mathcal{P}_m^{\mathrm{n}})$ connecting any two normal points in the same hinge set or two adjacent hinge sets (w.r.t. the total order) *hinge connections* and call the other edges *long range connections*. We describe the details in Section 4.

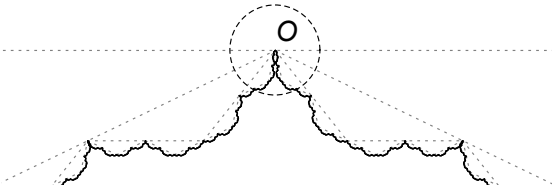
As we will see, all possible long range connections have a relatively simple form. Then, we show that we can break all long range connections by adding a set $\mathcal{P}_m^{\mathrm{a}}$ of *auxiliary points*. Each auxiliary point has a unique *center* which is the normal point closest to it. Let the minimum distance between any two normal points in $\mathcal{P}_m^{\mathrm{n}}$ be Δ. The distance between an auxiliary point and its center is much less than Δ. Naturally, we can extend the concepts of hinge set and long range connection to include the auxiliary points. An *extended hinge set* consists of the normal points in a hinge set and the auxiliary points centered on these normal points. We will see that the auxiliary points break all long range connections and



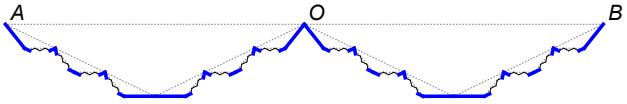
(a) Replace a horizontal segment by a sawteeth-like path.

(b) The lengths of the sawteeth-like paths are independent of the number of sawteeth.

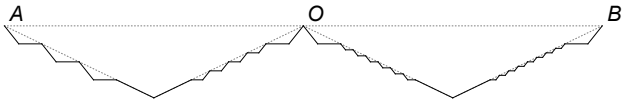
(c) Replace the segments by sawteeth-like path recursively.



(d) An enlarged view of Figure 1c around point O.



(e) Do not replace the bold segments further.



(f) The paths have different numbers of sawteeth and the sizes of sawteeth may not be the same.

Figure 1 The overview of the counterexample construction. Figure 1a-1f illustrate the fractal and its variants.

introduce no new long range connection. We describe the details in Section 5.

Finally, according to the process above, we can see that the shortest path between the normal points A and B in $\mathsf{YY}_{2k+1}(\mathcal{P}_m)$ for $m \in \mathbb{Z}^+$ should pass through all extended hinge sets in order. Thereby, the length of the shortest path between A and B diverges as m approaches infinity. See Section 5 for the details.

3 The positions of normal points

3.1 Some basic concepts

Let $k \geq 3$ be a fixed positive integer.[3] We consider YY_{2k+1} and let $\theta = 2\pi/(2k+1)$.

▶ **Definition 6** (Cone Boundary)**.** Consider any two points u and v. If the polar angle of $\overrightarrow{uv}$ is $j\theta = j \cdot 2\pi/(2k+1)$ for some integer $j \in [0, 2k]$, we call the ray $\overrightarrow{uv}$ a cone boundary for point u.

[3] Note that the cases $k = 1, 2$ have been proved in [15]

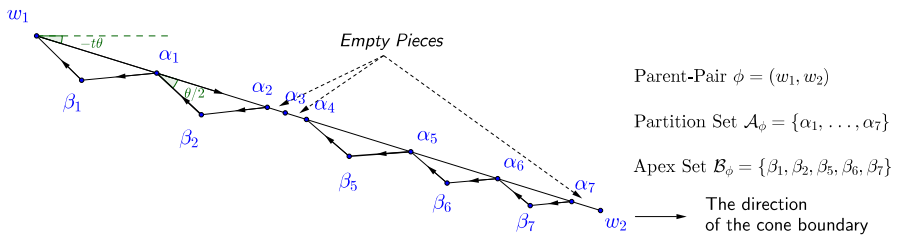


Figure 2 An example of one gadget. $\phi = (w_1, w_2)$ is the parent-pair in the gadget. $\mathcal{A}_\phi = \{\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_7\}$ is the partition set and $\mathcal{B}_\phi = \{\beta_1, \beta_2, \beta_5, \beta_6, \beta_7\}$ is the apex set. There are eight pieces, in which $w_1\alpha_1, \alpha_1\alpha_2, \alpha_4\alpha_5, \alpha_5\alpha_6, \alpha_6\alpha_7$ are non-empty pieces and $\alpha_2\alpha_3, \alpha_3\alpha_4, \alpha_7 w_2$ are empty pieces.

▶ **Property 7.** Consider two points u and v in $\mathcal{P}$. If $\overrightarrow{uv}$ consists of a cone boundary in $\mathsf{YY}_{2k+1}(\mathcal{P})$, its reverse $\overrightarrow{vu}$ is not a cone boundary.

In retrospect, this property is a key difference between odd and even Yao-Yao graphs, and our counterexample for odd Yao-Yao graphs will make crucial use of the property.

▶ **Definition 8** (Boundary Pair)**.** A boundary pair consists of two ordered points, denoted by (w_1, w_2), such that $\overrightarrow{w_1 w_2}$ is a cone boundary of point w_1.

For convenience, we refer to the word *pair* in the paper as the boundary pair defined in Definition 8. According to Property 7, if (w_1, w_2) is a pair, its reverse (w_2, w_1) is not a pair. Moreover, if a pair ϕ is $(u, \cdot)$ or $(\cdot, u)$, we say that the point u belongs to ϕ (i.e., $u \in \phi$).

Gadget One gadget G_ϕ consists of three groups of points. We explain them one by one. See Figure 2 for an example.

1. The first group is the pair $\phi = (w_1, w_2)$. We call ϕ the *parent-pair* of the gadget G_ϕ.
2. The second group is a set $\mathcal{A}_\phi$ of points on the segment of (w_1, w_2). We call the set $\mathcal{A}_\phi$ a *partition set* and call the points of $\mathcal{A}_\phi$ the *partition points* of ϕ. The set $\mathcal{A}_\phi$ divides the segment into $|\mathcal{A}_\phi| + 1$ parts, each we call a *piece* of the segment. There are two types of pieces. One is called an *empty piece* and the other a *non-empty piece*. Whether a piece is empty or not is determined in the process of the construction, which we will explain in Section 3.2.
3. For each non-empty piece, $\alpha_{i-1}\alpha_i$, we add a point β_i such that $\angle\alpha_{i-1}\beta_i\alpha_i = \pi - \theta$ and $|\alpha_{i-1}\beta_i| = |\beta_i\alpha_i|$. All β_is are on the same side of $w_1 w_2$. We call such a point β_i an *apex point* of (w_1, w_2). Let $\mathcal{B}_\phi$ be the set of apex points generated by ϕ, which is called the *apex set* of pair ϕ. $\mathcal{B}_\phi$ is the third group of points. For any empty piece, we do not add the corresponding apex point.

Consider a gadget $\mathsf{G}_\phi[\mathcal{A}_\phi, \mathcal{B}_\phi]$, where $\phi = (w_1, w_2)$. For any non-empty piece $\alpha_{i-1}\alpha_i$ and the corresponding apex point β_i, the rays $\overrightarrow{\beta_i\alpha_{i-1}}$ and $\overrightarrow{\alpha_i\beta_i}$ (note the order of the points) are cone boundaries according to their polar angles. Thus, each point $\beta_i \in \mathcal{B}_\phi$ induces two pairs (β_i, α_{i-1}) and (α_i, β_i). We call all pairs (β_i, α_{i-1}) and (α_i, β_i) induced by points in $\mathcal{B}_\phi$ the *child-pairs* of (w_1, w_2), and we say that they are *siblings* of each other. Note that there are some partition point which are not incident on any pair (e.g., α_3 in Figure 2). We call it an *isolated* point.

▶ **Definition 9** (The Order of the Child-pairs)**.** Consider a gadget $\mathsf{G}_{(w_1,w_2)}$. Suppose Φ is the set of the child-pairs of (w_1, w_2). Consider two pairs ϕ, φ in Φ. Define the order $\phi \prec \varphi$, if ϕ is closer to w_1 than φ. Here, the distance from a pair ϕ to a point w is the shortest distance from w to any point of ϕ.

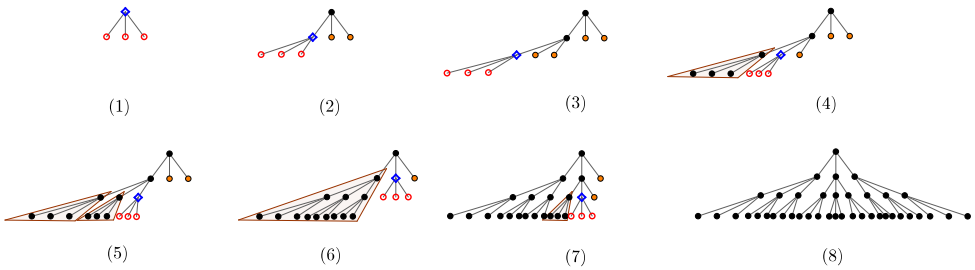


Figure 3 The process of generating a tree according to the DFS preorder. In each subfigure, ◇ represents a node we are visiting. The nodes generated in the step are denoted by ○. ● represents a node which has already been visited. ● represents a node which has been created but not visited yet. The nodes covered by light brown triangles are related to the projection process.

3.2 The construction

In this subsection, we construct an m-level tree. When the recursion level increases by 1, we need to replace each current pair by a gadget generated by the pair. See Figure 3 for an example. The recursion can be naturally represented as a tree $\mathcal{T}$. The tree is generated according to the DFS preorder, starting from the root. W.l.o.g, we assume that the root of $\mathcal{T}$ is (μ_1, μ_2) and $\mu_1\mu_2$ is horizontal. We call a pair *a leaf-pair* if it is a leaf node in the tree and *an internal-pair* otherwise. Each time when we visit an internal-pair, we generate its gadget by two steps which are called *projection* and *refinement* and explained in detail soon. Generating its gadget is equivalent to generating its child-pairs in $\mathcal{T}$ (we, however, do not visit those children during the generation. They will be visited later according to the DFS preorder). Whether a child-pair is a leaf or not is determined when the gadget is created. Hence, note that not all leaf-pairs are at *level-m*. We call the points generated by the process *normal points* and denote the set of these points by $\mathcal{P}^{\mathrm{n}}_m$ where m is the level of the tree and n represents the word "normal".

Root gadget. Let d_0 be a large positive constant integer (see Lemma 16 for detail). Consider a pair $\phi = (\mu_1, \mu_2)$. Let $\mathcal{A}_\phi$ be its partition set which contains points

$$\alpha_i = \mu_1 \cdot \frac{d_0 - i}{d_0} + \mu_2 \cdot \frac{i}{d_0}, i \in [1, d_0 - 1].$$

For convenience, let $\alpha_0 = \mu_1, \alpha_{d_0} = \mu_2$. The points in $\mathcal{A}_\phi$ partition the segment $\mu_1\mu_2$ into d_0 pieces with equal length $|\mu_1\mu_2|/d_0$. All pieces in the root gadget are non-empty. For each piece $\alpha_{i-1}\alpha_i$, we add an apex point β_i below $\mu_1\mu_2$. Let $\mathcal{B}_\phi = \{\beta_i\}_{i\in[1,d_0]}$ be the apex set.

Projection and refinement. The projection and refinement generate the partition points of pair ϕ. The purpose of the projection is to restrict all possible *long range connections* to a relatively simple form. See Section 4 for the details. The purpose of the refinement is to make the sibling pairs have relatively the same length, hence, make it possible to repeat the projection process recursively. Formally speaking, the refinement maintains the following property over the construction.

▶ **Property 10.** We call the segment connecting the two points of the pair the *segment of the pair* and call the length of that segment the *length of the pair*. Consider an internal-pair ϕ. Suppose φ is a sibling of ϕ. The length of pair φ is at least half of the length of pair ϕ.

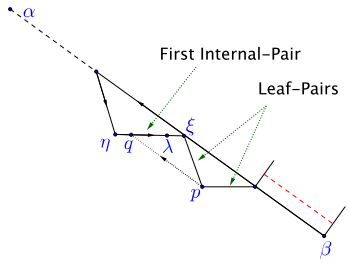
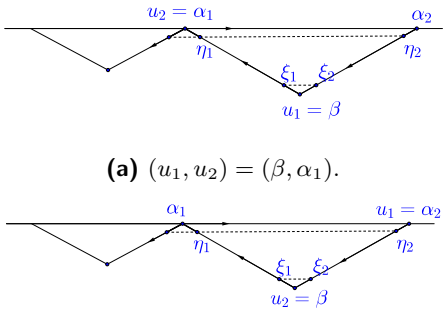


Figure 4 An example of the projection for the first internal-pair $\phi = (\eta, \xi)$.



(a) $(u_1, u_2) = (\beta, \alpha_1)$.

(b) $(u_1, u_2) = (\alpha_2, \beta)$.

Figure 5 The two cases for refinement.

– Projection. Consider a pair (β, α) with the set Φ being its child-pairs. We decide whether a pair in Φ is a leaf-pair or an internal-pair after introducing the process projection and refinement. There is a clear order between the leaf-pairs and the internal-pairs. We provide that the property of the order here and prove it in the full version.

▶ **Property 11.** Consider a pair (β, α) with the set Φ of its child-pairs. For $\phi_1, \phi_2, \phi_3 \in \Phi$, if $\phi_1 \prec \phi_2 \prec \phi_3$ and ϕ_1 and ϕ_3 are two internal-pairs, then ϕ_2 is an internal-pair.

Next, we apply the projection to an internal-pair $\phi \in \Phi$ since only internal-pairs have children. We explain the projection formally below. We define the first internal-pair in direction $\overrightarrow{\beta\alpha}$ as the *first internal-pair* of Φ. Let $\mathcal{T}_\phi$ be the set of points in the subtree rooted at ϕ. Depending on whether ϕ is the first internal-pair of Φ, there are two cases.

- Pair ϕ is the first internal-pair of Φ: In Figure 4, suppose pair $\phi = (\eta, \xi)$ is the first internal child-pair of (β, α) and the length of ϕ is δ. Point ξ is the partition point in ϕ. First, we add a point λ on the segment of ϕ such that $|\xi\lambda| = \delta/d_0$. Second, for each leaf-pair $\varphi \prec \phi$, project the apex point in φ to the segment of ϕ along the direction $\overrightarrow{\beta\alpha}$, [4] e.g., project p to q in Figure 4. Note that the length of leaf-pair φ is at least $\delta/2$ according to Property 10. Thus, there is no point between λ and ξ as long as $d_0 > 2$. Formally, we denote the operation by

$$\widehat{\mathcal{A}}_\phi \leftarrow \mathsf{Proj}\left[\bigcup\nolimits_{\varphi \prec \phi, \varphi \in \Phi} \mathcal{T}_\varphi\right] \cup \lambda. \tag{1}$$

- Pair ϕ is not the first internal-pair: According to the DFS preorder, we have already constructed the subtrees rooted at $\varphi \prec \phi$. We project all points $p \in \bigcup_{\varphi \prec \phi, \varphi \in \Phi} \mathcal{T}_\varphi$ to the segment of ϕ along the direction $\overrightarrow{\beta\alpha}$. Let the partition set $\widehat{\mathcal{A}}_\phi$ of ϕ be the set of the projected points falling inside the segment of ϕ. If several points overlap, we keep only one of them. Formally, we denote the operation by

$$\widehat{\mathcal{A}}_\phi \leftarrow \mathsf{Proj}\left[\bigcup\nolimits_{\varphi \prec \phi, \varphi \in \Phi} \mathcal{T}_\varphi\right]. \tag{2}$$

– Refinement. The partition points of ϕ divide the segment of ϕ into pieces. In order to maintain Property 10, we need to ensure that all non-empty pieces of ϕ have approximately

[4] If the projected point falls outside the segment of ϕ, we do not need to add a normal point.

the same length. We call this process the *refinement* operation. After the projection, we obtain a candidate partition set $\widehat{\mathcal{A}}_\phi$ of ϕ (defined in (1) and (2)). However, note that the length between the pieces may differ a lot.

W.l.o.g., suppose pair ϕ has unit length, $|\widehat{\mathcal{A}}_\phi| = n$ and $n > d_0$.[5] Suppose $\phi = (u_1, u_2)$. We distinguish two cases based on whether the first point u_1 is a partition point or an apex point.

- If u_1 is an apex point, we mark the piece incident on u_2. See Figure 5a for an illustration, in which $(u_1, u_2) = (\beta, \alpha_1)$ and piece $\alpha_1\eta_1$ is the marked piece.
- If u_1 is a partition point, we mark the pieces incident on u_1 and u_2. See Figure 5b for an illustration, in which $(u_1, u_2) = (\alpha_2, \beta)$ and piece $\alpha_2\eta_2$ and $\xi_2\beta$ are the marked pieces.

We do not add any point in the marked pieces under refinement. Consider two sibling pairs (β, α_1) and (α_2, β) where β is an apex point. Suppose $\alpha_1\eta_1, \alpha_2\eta_2, \xi_2\beta$ are the marked pieces of the two pairs and ξ_1 is the point on the segment $\beta\alpha_1$ which is projected to ξ_2.[6] Then $|\beta\xi_1| = |\beta\xi_2|$ and $|\alpha_1\eta_1| = |\alpha_2\eta_2|$ after the refinement. See Figure 5 for an example.

Denote the length of the ith piece (defined by $\widehat{\mathcal{A}}_\phi$) by δ_i. Let $\delta_o = 1/n^2$. Except for the marked pieces, for each piece which is at least twice longer than δ_o, we place $\lfloor \delta_i/\delta_o \rfloor - 1$ equidistant points on the piece, which divide the piece into $\lfloor \delta_i/\delta_o \rfloor$ equal-length parts.

We call this process the *refinement* and denote the resulting point set by

$$\mathcal{A}_\phi \leftarrow \mathsf{Refine}[\widehat{\mathcal{A}}_\phi]. \tag{3}$$

Note that The number of points added in the refinement process is at most $O(n^2)$ since the segment of pair ϕ has unit length and $\delta_o \geq 1/n^2$. We call each piece whose length is less than δ_o a *short piece*. The short pieces remain unchanged before and after the refinement. Moreover, the refinement does not introduce any new short piece for the pair. In the full version, we prove that the sum of lengths of the short pieces is less than $1/d_0$.

Deciding Emptiness, Leaf-Pairs and Internal-Pairs We defer the details to the full version and just give the definitions here. In the full version, we will prove Property 10 and provide more properties of the construction.

Consider a pair ϕ whose apex point is β and partition point is α.[7] We let the piece incident on the apex point β and the short pieces be empty and the other pieces be non-empty.

For each non-empty piece, we generate one apex point. As we have discussed before, the apex set $\mathcal{B}_\phi$ induces the set Φ of child-pairs of ϕ. Let the three pairs closest to α and two pairs closest to β be *leaf-pairs*. We do not further expand the tree from the leaf-pairs. Let the other pairs be the *internal-pairs*.

Overall, after the projection and refinement process, we can generate the gadget for any pair in the tree. We denote this process by

$$\mathsf{G}_\phi \leftarrow \mathsf{Proj\text{-}Refn}(\phi). \tag{4}$$

In the full version, we prove that the Property 10 and 11 hold under the construction and provide some other properties about the construction.

[5] If $n \leq d_0$, we repeatedly split the inner pieces (i.e., all pieces except for the two pieces incident on the points of ϕ) into two equal-length pieces until the number of the points in $\widehat{\mathcal{A}}_\phi$ is larger than d_0.

[6] Point ξ_1 must exist since ξ_2 is a projected point and there is no point in the marked piece $\xi_2\beta$.

[7] Note that the first point of a pair can be either apex point or partition point. Here, $\phi = (\alpha, \beta)$ or $\phi = (\beta, \alpha)$ depending on whether first point of ϕ is apex point or not.

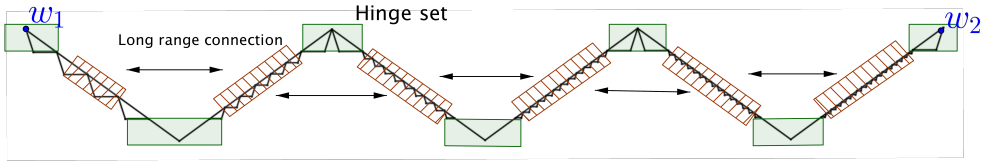
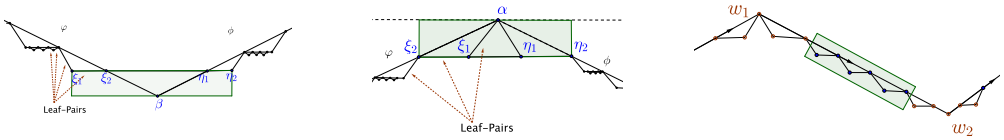


Figure 6 The overview of the hinge set decomposition. Roughly speaking, each set of points covered by a green rectangle ▭ is a hinge set. Recursively, we can further decompose the points covered by shadowed rectangle ▨ into hinge sets.



(a) The hinge set centered on an apex point.

(b) The hinge set centered on a partition point.

(c) The hinge set consisting of the leaf-pairs at *level-m*.

Figure 7 The hinge sets centered on a point in an internal-pair.

4 Hinge set decomposition of the normal points

We decompose $\mathcal{P}_m^{\mathrm{n}}$ into a collection of sets of points such that each normal point belongs to exactly one set. We call these sets *hinge sets*. Briefly speaking, each hinge set is a set of points which are close geometrically. See Figure 6 for an overview. Consider a pair $\hat{\phi}$ at *level-l*, $l < m-1$ with partition point set $\mathcal{A}_{\hat{\phi}}$ and apex point set $\mathcal{B}_{\hat{\phi}}$. Let $\hat{\Phi}$ be the set of the child-pairs of $\hat{\phi}$. Formally, the hinge sets are defined as follows.

- The hinge set centered on a point $\beta \in \mathcal{B}_{\hat{\phi}}$ which belongs to one or two internal-pairs in $\hat{\Phi}$: We denote the two pairs by φ and ϕ. The hinge set centered at β includes: β itself, the child-pair of φ closest to β (denoted by (ξ_1, ξ_2) in Figure 7a) and the child-pair of ϕ closest to β (denoted by (η_1, η_2) in Figure 7a).[8]
- The hinge set centered on a point $\alpha \in \mathcal{A}_{\hat{\phi}}$ which belongs to one or two internal-pairs in $\hat{\Phi}$: We denote the two pairs by φ and ϕ. The hinge set centered on α includes: α itself, the two child-pairs closest to α of φ and ϕ respectively (denoted by (ξ_2, ξ_1) and (η_1, η_2) in Figure 7b).[9]

W.l.o.g., we process the points μ_1 and μ_2 in the root pair in the same way as the partition points in $\mathcal{A}_{(\mu_1,\mu_2)}$. So far, some points at *level-m* still do not belong to any hinge set.

- The hinge set consisting of the leaf-pairs at *level-m*: Consider any pair $\phi = (w_1, w_2)$ at *level-*$(m-1)$. Define the set difference of $\mathcal{A}_\phi \cup \mathcal{B}_\phi$ and the hinge sets centered on w_1 and w_2 as a hinge set.[10] See Figure 7c.

Overall, we decompose the points $\mathcal{P}_m^{\mathrm{n}}$ into a collection of hinge sets.

▶ **Lemma 12.** *Each point p in $\mathcal{P}_m^{\mathrm{n}}$ belongs to exactly one hinge set.*

[8] Note that $\xi_1, \xi_2, \eta_1, \eta_2$ only belong to leaf-pairs.

[9] There is a degenerated case in which α is an isolated point which does not belong to any internal-pair in $\hat{\Phi}$. We process the case in the full version.

[10] Although these points form the leaf-pairs at *level-m*, these leaf-pairs are the "candidate internal-pairs" to generate the points at *level-*$(m+1)$.

Order of the hinge sets. We define the total order of all hinge sets. We denote the order by "$\prec_h$", which is different from the previous order "$\prec$". The $\prec_h$ is in fact consistent with the order of traversing the fractal path from μ_1 to μ_2. Rigorously, we define $\prec_h$ in the full version.

▶ **Definition 13** (Long range connection)**.** We call an edge connecting two points in two non-adjacent hinge sets a *long range connection*.

If there is no long range connection, the total order of the hinge sets corresponds to the ordering of the shortest path from μ_1 to μ_2 in the final construction. It means that each hinge set has at least one point on the shortest path between μ_1 and μ_2 and the order of these points is consistent with $\prec_h$. However, there indeed exist long range connections among normal points. Fortunately, the long range connections in $\mathsf{YY}_{2k+1}(\mathcal{P}_m^{\mathrm{n}})$ have relatively simple form. After introducing some auxiliary points (in Section 5), we can cut the long range connections without introducing any new long range connections.

Now, we examine long range connections in $\mathsf{YY}_{2k+1}(\mathcal{P}_m^{\mathrm{n}})$. We give the sketch below and defer the details to the full version. Intuitively, these properties result from Property 7 which does not hold for even Yao-Yao graphs.

- First, we claim that if we can cut all long range connections between the points in $\mathcal{T}_\phi$ and $\mathcal{T}_\varphi$ for any two sibling pairs ϕ and φ, then there is no long range connection.
- Then, we consider the long range connections between two subtrees of any two sibling pairs. Consider two sibling pairs ϕ and φ with subtrees $\mathcal{T}_\phi$ and $\mathcal{T}_\varphi$ respectively. Suppose p belongs to $\mathcal{T}_\phi$ and q belongs to $\mathcal{T}_\varphi$. We prove that if directed edge $\overrightarrow{pq}$ is in $\mathsf{YY}_{2k+1}(\mathcal{P}_m^{\mathrm{n}})$, then $\phi \prec \varphi$.
- Finally, we consider the possible long range connection from $\mathcal{T}_\phi$ to $\mathcal{T}_\varphi$ for $\phi \prec \varphi$. Suppose p belongs to $\mathcal{T}_\phi$. Note that the points of $\mathcal{T}_\varphi$ are located in at most two cones of p based on Property 10. We prove that for each point p, only one of the two cones may contain a long range connection. Moreover, there exists a cone boundary pu such that (1) the long range connection is on the cone boundary if pu is in the cone, (2) or there is a point $q \in \mathsf{G}_\varphi$ on pu and $|pq| < |pv|$ for any point v in the cone if pu is not in the cone.[11]

5 The positions of auxiliary points

We discuss how to use the auxiliary points to cut the long range connections in the Yao-Yao graph $\mathsf{YY}_{2k+1}(\mathcal{P}_m^{\mathrm{n}})$. Denote the set of auxiliary points by $\mathcal{P}_m^{\mathrm{a}}$. Let $\mathcal{P}_m = \mathcal{P}_m^{\mathrm{n}} \cup \mathcal{P}_m^{\mathrm{a}}$. First, we consider a simple example to see how auxiliary points work. Consider three points u, v and w. Line uv is horizontal, and $\angle wvu = \angle wuv = \theta/2$. The point ξ_1 and ξ_2 are two points on segment uw and vw respectively. $\xi_1\xi_2$ is horizontal. See Figure 8. Note that the polar angles of a cone in the Yao-Yao graph belong to a left side half-open interval in the counterclockwise direction. Thus, uv is in the YY_{2k+1} graph, which is the shortest path between u and v. However, we can add an auxiliary point r close to v and $\angle rvu < \theta/2$. Then, the shortest path between uv becomes $u\xi_1\xi_2rv$.

The positions of the auxiliary points Let Δ be the minimum distance between any two normal points and n be the number of the normal points. Recall that we partition the root pair μ_1, μ_2 into d_0 equidistant pieces. Let γ be a very small angle, such as $\gamma = \theta d_0^{-1}$. Let

[11] Since a cone is a half-open interval in the counterclockwise direction, each cone has one boundary outside the cone.

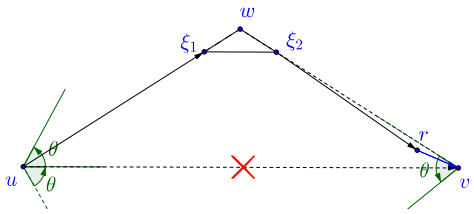


Figure 8 A simple example to explain how an auxiliary point cuts a long range connection.

$\sigma = \max\{\sin(\theta/2-\gamma)/\sin\gamma, \sin^{-1}(\theta/2-\gamma)\} + \epsilon$ for some small $\epsilon > 0$. Let $\chi = d_0\sigma^n\Delta^{-1}$. Roughly speaking, $\chi \gg d_0 > \sigma > 1$.

Now, we explain the positions of the auxiliary points formally. Inspired by the example in Figure 8, we call the normal point closest to an auxiliary point the *center* of the auxiliary point. Then, we find the *candidate centers* to add auxiliary points.

▶ **Definition 14** (Candidate center)**.** Consider a pair (v_1, v_2) and a set Φ of its child-pairs. $\phi, \varphi \in \Phi$ and $\varphi \prec \phi$. p is a point in $\mathcal{T}_\varphi$ and q is the projected point of p in $\mathcal{T}_\phi$. $r \in \mathsf{G}_\phi$ is the closest point to p on segment pq.[12] We call r a *candidate center* of auxiliary points.

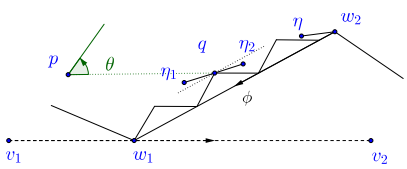
We traverse $\mathcal{T}$ in the DFS preorder. Each time we reach a pair ϕ, we find all candidate centers in G_ϕ and add auxiliary points centered on them.[13] Moreover, let the order of ϕ in the DFS preorder w.r.t. $\mathcal{T}$ be κ. The distance between the auxiliary point and its center just depends on κ. Let $\phi = (w_2, w_1)$ and (v_1, v_2) be the parent-pair of ϕ. There are two cases according to $\angle(v_1v_2, w_1w_2) = \theta/2$ or $-\theta/2$.

- $\angle(v_1v_2, w_1w_2) = \theta/2$ (see Figure 9a and 9b):
  - If $q = w_1$, do not add auxiliary point.
  - If $q = w_2$, we add the point η such that $\angle(w_2w_1, w_2\eta) = -\gamma$ and $|w_2\eta| = \sigma^\kappa\chi^{-1}$.
  - Otherwise, we add two points η_1 and η_2 centered on q such that $\angle(w_2w_1, q\eta_1) = \angle(w_2w_1, \eta_2q) = -\gamma$ and $|q\eta_1| = |\eta_2q| = \sigma^\kappa\chi^{-1}$.
- $\angle(v_1v_2, w_1w_2) = -\theta/2$ (see Figure 9c and 9d):
  - If $q = w_1$, do not add auxiliary point.
  - If $q = w_2$, we add the point η such that $\angle(w_2w_1, w_2\eta) = \gamma$ and $|w_2\eta| = \sigma^\kappa\chi^{-1}$.
  - If p and q are in the same hinge set (i.e., p, q are the points ξ_1, ξ_2 in Figure 9c or 9d), we add two points η_1 and η_2 centered on q such that $\angle(w_2w_1, q\eta_1) = \angle(w_2w_1, \eta_2q) = \gamma$ and $|q\eta_1| = |\eta_2q| = \sigma^\kappa\chi^{-1} + \epsilon_0$ where ϵ_0 is much less than the distance between any two points in $\mathcal{P}_m$.[14]
  - Otherwise, we add two points η_1 and η_2 centered on q such that $\angle(w_2w_1, q\eta_1) = \angle(w_2w_1, \eta_2q) = \gamma$ and $|q\eta_1| = |\eta_2q| = \sigma^\kappa\chi^{-1}$.

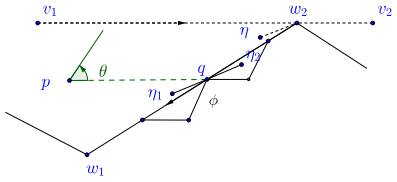
Let *extended hinge set* consist of the normal points in the hinge set and the auxiliary points centered on these normal points. The concept of long range connections can be extended to the extended hinge sets. Then, there is no long range connection in $\mathsf{YY}_{2k+1}(\mathcal{P}_m)$.

▶ **Lemma 15.** *There is no long range connection in* $\mathsf{YY}_{2k+1}(\mathcal{P}_m)$.

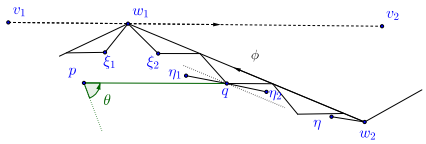
[12] r may not be q. See the full version for the proof of the existence of r.
[13] Note that the candidate centers belong to G_ϕ, may not belong to ϕ itself.
[14] It is slightly different from the first case. We add two auxiliary points with distance slightly larger than $\sigma^\kappa\chi^{-1}$. The reason is that the cone is half-open half-close in the counterclockwise direction. It will help a lot to unify the proof in the same framework.



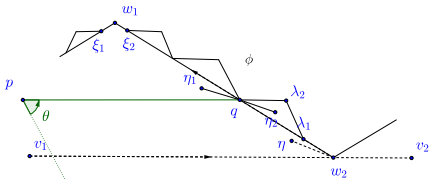
(a) Case 1a: w_1 is the partition point and ϕ is on the left side of v_1v_2.



(b) Case 1b: w_1 is the apex point and ϕ is on the right side of v_1v_2.



(c) Case 2a: w_1 is the partition point and ϕ is on the right side of v_1v_2.



(d) Case 2b: w_1 is the apex point and ϕ is on the left side of v_1v_2.

Figure 9 The auxiliary points for each point. Here $\phi = (w_2, w_1)$ and $q \in \mathsf{G}_\phi$. η_1 and η_2 are two auxiliary points centered on q. Note that $|\eta_1 q|$ and $|\eta_2 q|$ are very small in fact.

We defer the detailed proof to the full version. First, we prove that if for any two sibling pairs ϕ and φ in $\mathcal{T}$ at *level-l* for $l \leq m-1$, there is no long range connection between the points in $\mathcal{T}_\phi$ and $\mathcal{T}_\varphi$, then there is no long range connection (see Claim 17 in the full version). Next, for any $l \leq m-1$, we examine any two points p and q which belong to $\mathcal{T}_\phi$ and $\mathcal{T}_\varphi$ respectively, where ϕ and φ are two sibling pairs at *level-l* . We give a necessary condition when there is a edge pq in $\mathsf{YY}_{2k+1}(\mathcal{P}_m)$ (see Lemma 25 in the full version). We also provide some useful properties about the construction (see Property 24 in the full version). These properties indicate that for any possible long range connection, we can always find a point r (see point r in Figure 8 for an example) to break the connection (see Lemma 26 in the full version).

Finally, we can prove that the shortest paths between μ_1 and μ_2 in the root pair should pass through all extended hinge sets in order, hence, diverges as m approaches infinity.

▶ **Lemma 16.** *The length of the shortest path between μ_1 and μ_2 in $\mathsf{YY}_{2k+1}(\mathcal{P}_m)$ for $k \geq 3$ is at least ρ^m, for some $\rho = (1-O(d_0^{-1}))\cdot\cos^{-1}(\theta/2)$. Thus, by setting $d_0 > \lceil 6(1-\cos(\theta/2))^{-1}\rceil$, the length diverges as m approaches infinity.*

References

1 Franz Aurenhammer. Voronoi diagrams – survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3):345–405, 1991.

2 Luis Barba, Prosenjit Bose, Mirela Damian, Rolf Fagerberg, Wah Loon Keng, Joseph O'Rourke, André van Renssen, Perouz Taslakian, Sander Verdonschot, and Ge Xia. New and improved spanning ratios for Yao graphs. *JoCG*, 6(2):19–53, 2015.

3 Luis Barba, Prosenjit Bose, Jean-Lou De Carufel, André van Renssen, and Sander Verdonschot. On the stretch factor of the Θ_4-graph. In *Workshop on Algorithms and Data Structures*, pages 109–120. Springer, 2013.

4 Matthew Bauer and Mirela Damian. An infinite class of sparse-Yao spanners. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 184–196. SIAM, 2013.

5 Nicolas Bonichon, Cyril Gavoille, Nicolas Hanusse, and David Ilcinkas. Connections between Θ-graphs, Delaunay triangulations, and orthogonal surfaces. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 266–278. Springer, 2010.

6 Prosenjit Bose, Mirela Damian, Karim Douïeb, Joseph O'rourke, Ben Seamone, Michiel Smid, and Stefanie Wuhrer. $\pi/2$-angle Yao graphs are spanners. *International Journal of Computational Geometry & Applications*, 22(01):61–82, 2012.

7 Prosenjit Bose, Anil Maheshwari, Giri Narasimhan, Michiel Smid, and Norbert Zeh. Approximating geometric bottleneck shortest paths. *Computational Geometry*, 29(3):233–249, 2004.

8 Prosenjit Bose, Pat Morin, André van Renssen, and Sander Verdonschot. The Θ_5-graph is a spanner. *Computational Geometry*, 48(2):108–119, 2015.

9 Prosenjit Bose, André van Renssen, and Sander Verdonschot. On the spanning ratio of theta-graphs. In *Workshop on Algorithms and Data Structures*, pages 182–194. Springer, 2013.

10 Paul Chew. There is a planar graph almost as good as the complete graph. In *Proceedings of the second annual symposium on Computational geometry*, pages 169–177. ACM, 1986.

11 Mirela Damian. A simple Yao-Yao-based spanner of bounded degree. *arXiv preprint arXiv:0802.4325*, 2008.

12 Mirela Damian, Nawar Molla, and Val Pinciu. Spanner properties of $\pi/2$-angle Yao graphs. In *Proc. of the 25th European Workshop on Computational Geometry*, pages 21–24. Citeseer, 2009.

13 Mirela Damian and Naresh Nelavalli. Improved bounds on the stretch factor of Y_4. *Computational Geometry*, 62:14–24, 2017.

14 Mirela Damian and Kristin Raudonis. Yao graphs span Θ-graphs. In *Combinatorial Optimization and Applications*, pages 181–194. Springer, 2010.

15 Nawar M El Molla. Yao spanners for wireless ad hoc networks. Master's thesis, Villanova University, 2009.

16 David Eppstein. Spanning trees and spanners. *Handbook of computational geometry*, pages 425–461, 1999.

17 David Eppstein. Beta-skeletons have unbounded dilation. *Computational Geometry*, 23(1):43–52, 2002.

18 BE Flinchbaugh and LK Jones. Strong connectivity in directional nearest-neighbor graphs. *SIAM Journal on Algebraic Discrete Methods*, 2(4):461–463, 1981.

19 K Ruben Gabriel and Robert R Sokal. A new statistical approach to geographic variation analysis. *Systematic Biology*, 18(3):259–278, 1969.

20 Matthias Grünewald, Tamás Lukovszki, Christian Schindelhauer, and Klaus Volbert. Distributed maintenance of resource efficient wireless network topologies. In *European Conference on Parallel Processing*, pages 935–946. Springer, 2002.

21 Lujun Jia, Rajmohan Rajaraman, and Christian Scheideler. On local algorithms for topology control and routing in ad hoc networks. In *Proceedings of the fifteenth annual ACM symposium on Parallel algorithms and architectures*, pages 220–229. ACM, 2003.

22 Iyad A Kanj and Ge Xia. On certain geometric properties of the Yao-Yao graphs. In *Combinatorial Optimization and Applications*, pages 223–233. Springer, 2012.

23 Jian Li and Wei Zhan. Almost all even Yao-Yao graphs are spanners. *The 24rd Annual European Symposium on Algorithms*, 2016.

24 Xiang-Yang Li. *Wireless Ad Hoc and Sensor Networks: Theory and Applications*. Cambridge, 6 2008.

25 Xiang-Yang Li, Peng-Jun Wan, and Yu Wang. Power efficient and sparse spanner for wireless ad hoc networks. In *Computer Communications and Networks, 2001. Proceedings. Tenth International Conference on*, pages 564–567. IEEE, 2001.

26 Xiang-Yang Li, Peng-Jun Wan, Yu Wang, and Ophir Frieder. Sparse power efficient topology for wireless networks. In *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on*, pages 3839–3848. IEEE, 2002.

27 Giri Narasimhan and Michiel Smid. *Geometric spanner networks*. Cambridge University Press, 2007.

28 Jim Ruppert and Raimund Seidel. Approximating the d-dimensional complete euclidean graph. In *Proceedings of the 3rd Canadian Conference on Computational Geometry (CCCG 1991)*, pages 207–210, 1991.

29 Jörg-Rüdiger Sack and Jorge Urrutia. *Handbook of computational geometry*. Elsevier, 1999.

30 Christian Schindelhauer, Klaus Volbert, and Martin Ziegler. Spanners, weak spanners, and power spanners for wireless networks. In *International Symposium on Algorithms and Computation*, pages 805–821. Springer, 2004.

31 Christian Schindelhauer, Klaus Volbert, and Martin Ziegler. Geometric spanners with applications in wireless networks. *Computational Geometry*, 36(3):197–214, 2007.

32 Godfried T Toussaint. The relative neighbourhood graph of a finite planar set. *Pattern recognition*, 12(4):261–268, 1980.

33 Yu Wang, Xiang-Yang Li, and Ophir Frieder. Distributed spanners with bounded degree for wireless ad hoc networks. *International Journal of Foundations of Computer Science*, 14(02):183–200, 2003.

34 Andrew Chi-Chih Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM Journal on Computing*, 11(4):721–736, 1982.

Deletion in Abstract Voronoi Diagrams in Expected Linear Time

Kolja Junginger
Faculty of Informatics, USI Università della Svizzera italiana,
Lugano, Switzerland
kolja.junginger@usi.ch

Evanthia Papadopoulou
Faculty of Informatics, USI Università della Svizzera italiana,
Lugano, Switzerland
evanthia.papadopoulou@usi.ch

Abstract

Updating an abstract Voronoi diagram in linear time, after deletion of one site, has been an open problem for a long time. Similarly for various concrete Voronoi diagrams of generalized sites, other than points. In this paper we present a simple, expected linear-time algorithm to update an abstract Voronoi diagram after deletion. We introduce the concept of a *Voronoi-like diagram*, a relaxed version of a Voronoi construct that has a structure similar to an abstract Voronoi diagram, without however being one. Voronoi-like diagrams serve as intermediate structures, which are considerably simpler to compute, thus, making an expected linear-time construction possible. We formalize the concept and prove that it is robust under an insertion operation, thus, enabling its use in incremental constructions.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Abstract Voronoi diagram, linear-time algorithm, update after deletion, randomized incremental algorithm

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.50

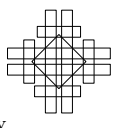
Related Version A full version of this paper is available at `http://arxiv.org/abs/1803.05372`.

Funding Supported in part by the Swiss National Science Foundation, DACH project SNF-200021E-154387

1 Introduction

The Voronoi diagram of a set S of n simple geometric objects, called sites, is a well-known geometric partitioning structure that reveals proximity information for the input sites. Classic variants include the *nearest-neighbor*, the *farthest-site*, and the *order-k* Voronoi diagram of S ($1 \leq k < n$). *Abstract Voronoi diagrams* [11] offer a unifying framework for various concrete and well-known instances. Some classic Voronoi diagrams have been well investigated, with optimal construction algorithms available in many cases, see e.g., [2] for references and more information or [16] for numerous applications.

For certain *tree-like* Voronoi diagrams in the plane, linear-time construction algorithms have been well-known to exist, see e.g., [1, 7, 13, 8]. The first technique was introduced by Aggarwal et al. [1] for the Voronoi diagram of points in convex position, given the order of points along their convex hull. It can be used to derive linear-time algorithms for other fundamental problems: (1) updating a Voronoi diagram of points after deletion of one site in

34th International Symposium on Computational Geometry (SoCG 2018).
Editors: Bettina Speckmann and Csaba D. Tóth; Article No. 50; pp. 50:1–50:14


Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

time linear to the number of the Voronoi neighbors of the deleted site; (2) computing the farthest Voronoi diagram of point-sites in linear time, after computing their convex hull; (3) computing the order-$(k+1)$ subdivision within an order-k Voronoi region. There is also a much simpler randomized approach for the same problems introduced by Chew [7]. Klein and Lingas [13] adapted the linear-time framework [1] to abstract Voronoi diagrams, under restrictions, showing that a *Hamiltonian abstract Voronoi diagram* can be computed in linear time, given the order of Voronoi regions along an unbounded simple curve, which visits each region *exactly once* and can intersect each bisector only *once*. This construction has been extended recently to include forest structures [4] under similar conditions, where no region can have multiple faces within the domain enclosed by a curve. The medial axis of a simple polygon is another well-known problem to admit a linear-time construction, shown by Chin et al. [8].

In this paper we consider the fundamental problem of updating a two-dimensional Voronoi diagram, after deletion of one site, and provide an expected linear-time algorithm to achieve this task. We consider the framework of abstract Voronoi diagrams to simultaneously address the various concrete instances under their umbrella. To the best of our knowledge, no linear-time construction algorithms are known for concrete diagrams of non-point sites, nor for abstract Voronoi diagrams. Related is our expected linear-time algorithm for the concrete farthest-segment Voronoi diagram [10][1], however, definitions are geometric, relying on star-shapeness and visibility properties of segment Voronoi regions, which do not extend to the abstract model. In this paper we consider a new formulation.

Abstract Voronoi diagrams. Abstract Voronoi diagrams (AVDs) were introduced by Klein [11]. Instead of sites and distance measures, they are defined in terms of bisecting curves that satisfy some simple combinatorial properties. Given a set S of n abstract sites, the bisector $J(p,q)$ of two sites $p,q \in S$ is an unbounded Jordan curve, homeomorphic to a line, that divides the plane into two open domains: the *dominance region of* p, $D(p,q)$ (having label p), and the *dominance region of* q, $D(q,p)$ (having label q), see Figure 1. The *Voronoi region* of p is

$$\mathrm{VR}(p,S) = \bigcap_{q \in S \setminus \{p\}} D(p,q).$$

The *(nearest-neighbor) abstract Voronoi diagram* of S is $\mathcal{V}(S) = \mathbb{R}^2 \setminus \bigcup_{p \in S} \mathrm{VR}(p,S)$.

Following the traditional model of abstract Voronoi diagrams (see e.g. [11, 3, 6, 5]) the system of bisectors is assumed to satisfy the following axioms, for every subset $S' \subseteq S$:

- **(A1)** Each nearest Voronoi region $\mathrm{VR}(p,S')$ is non-empty and pathwise connected.
- **(A2)** Each point in the plane belongs to the closure of a nearest Voronoi region $\mathrm{VR}(p,S')$.
- **(A3)** After stereographic projection to the sphere, each bisector can be completed to a Jordan curve through the north pole.
- **(A4)** Any two bisectors $J(p,q)$ and $J(r,t)$ intersect transversally and in a finite number of points. (It is possible to relax this axiom, see [12]).

$\mathcal{V}(S)$ is a plane graph of structural complexity $O(n)$ and its regions are simply-connected. It can be computed in time $O(n \log n)$, randomized [14] or deterministic [11]. To update $\mathcal{V}(S)$, after deleting one site $s \in S$, we compute $\mathcal{V}(S \setminus \{s\})$ within $\mathrm{VR}(s,S)$. The sequence of

[1] A preliminary version contains a gap when considering the linear-time framework of [1], thus, a linear-time construction for the farthest segment Voronoi diagram remains an open problem.

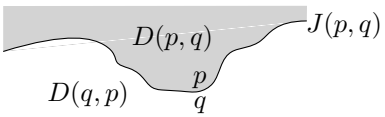


Figure 1 A bisector $J(p,q)$ and its dominance regions; $D(p,q)$ is shown shaded.

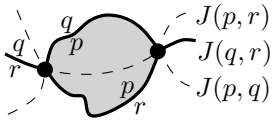


Figure 2 The Voronoi diagram $\mathcal{V}(\{p,q,r\})$ in solid lines. The shaded region is $\mathrm{VR}(p,\{p,q,r\})$.

site-occurrences along $\partial\mathrm{VR}(s,S)$ forms a Davenport-Schinzel sequence of order 2 and this constitutes a major difference from the respective problem for points, where no repetition can occur. $\mathcal{V}(S\setminus\{s\})\cap\mathrm{VR}(s,S)$ contains disconnected Voronoi regions, which introduce several complications. For example, $\mathcal{V}(S')\cap\mathrm{VR}(s,S'\cup\{s\})$ for $S'\subset S\setminus\{s\}$ may contain various faces that are not related to $\mathcal{V}(S\setminus\{s\})\cap\mathrm{VR}(s,S)$, and conversely, an arbitrary sub-sequence of $\partial\mathrm{VR}(s,S)$ need not correspond to any Voronoi diagram. At first sight, a linear-time algorithm may seem infeasible.

Our results. In this paper we give a simple randomized algorithm to compute $\mathcal{V}(S\setminus\{s\})$ within $\mathrm{VR}(s,S)$ in expected time linear on the complexity of $\partial\mathrm{VR}(s,S)$. The algorithm is simple, not more complicated than its counterpart for points [7], and this is achieved by computing simplified intermediate structures that are interesting in their own right. These are *Voronoi-like* diagrams, having a structure similar to an abstract Voronoi diagram, however, they are not Voronoi structures. *Voronoi-like regions* are supersets of real Voronoi regions, and their boundaries correspond to *monotone paths* in the relevant system of bisectors, rather than to an *envelope* in the same system as in a real Voronoi diagram (see Definition 5). We prove that Voronoi-like diagrams are well-defined, and also they are robust under an *insertion operation*, thus, making possible a randomized incremental construction for $\mathcal{V}(S\setminus\{s\})\cap\mathrm{VR}(s,S)$ in linear time. We expect the concept to find uses in other Voronoi computations, where computing intermediate relaxed structures may simplify the entire computation. A first candidate in this direction is the linear-time framework of Aggarwal et al. [1] that we plan to investigate next.

Our approach can be adapted (in fact, simplified) to compute in expected linear time the farthest abstract Voronoi diagram, after the sequence of its faces at infinity is known. The latter sequence can be computed in time $O(n\log n)$. We also expect that our algorithm can be adapted to compute the order-$(k{+}1)$ subdivision within an order-k abstract Voronoi region in expected time linear on the complexity of the region boundary.[2] Our technique can be applied to concrete diagrams that may not strictly fall under the AVD model such as Voronoi diagrams of line segments that may intersect and of planar straight-line graphs (including simple and non-simple polygons).

2 Preliminaries

Let S be a set of n abstract *sites* (a set of indices) that define an *admissible* system of bisectors in the plane $\mathcal{J}=\{J(p,q):p\neq q\in S\}$, which fulfills axioms (A1)–(A4) for every $S'\subseteq S$. The (nearest) Voronoi region of p is $\mathrm{VR}(p,S)=\bigcap_{q\in S\setminus\{p\}}D(p,q)$ and the Voronoi diagram of S is $\mathcal{V}(S)=\mathbb{R}^2\setminus\bigcup_{p\in S}\mathrm{VR}(p,S)$, see, e.g., Figure 2.

[2] The adaptation is non-trivial, thus, we only make a conjecture here and plan to consider details in subsequent work.

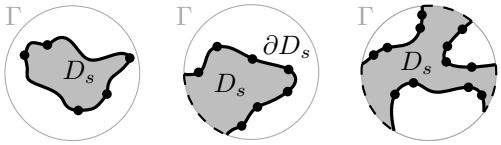


Figure 3 The domain $D_s = \mathrm{VR}(s, S) \cap D_\Gamma$.

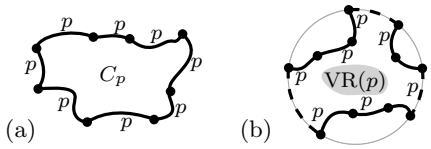


Figure 4 (a) A p-inverse cycle. (b) A p-cycle.

Bisectors that have a site p in common are called *p-related* or simply *related*; related bisectors can intersect at most twice [11, Lemma 3.5.2.5]. When two related bisectors $J(p, q)$ and $J(p, r)$ intersect, bisector $J(q, r)$ also intersects with them at the same point(s) [11], and these points are the Voronoi vertices of $\mathcal{V}(\{p, q, r\})$, see Figure 2. Since any two related bisectors in $\mathcal{J}$ intersect at most twice, the sequence of site occurrences along $\partial\mathrm{VR}(p, S)$, $p \in S$, forms a Davenport-Schinzel sequence of order 2 (by [19, Theorem 5.7]).

To update $\mathcal{V}(S)$ after deleting one site $s \in S$, we compute $\mathcal{V}(S \setminus \{s\})$ within $\mathrm{VR}(s, S)$, i.e., compute $\mathcal{V}(S \setminus \{s\}) \cap \mathrm{VR}(s, S)$. Its structure is given in the following lemma. Figure 7(a) illustrates $\mathcal{V}(S \setminus \{s\}) \cap \mathrm{VR}(s, S)$ (in red) for a bounded region $\mathrm{VR}(s, S)$, where the region's boundary is shown in bold.

▶ **Lemma 1.** *$\mathcal{V}(S \setminus \{s\}) \cap VR(s, S)$ is a forest having exactly one face for each Voronoi edge of $\partial VR(s, S)$. Its leaves are the Voronoi vertices of $\partial VR(s, S)$, and points at infinity if $VR(s, S)$ is unbounded. If $VR(s, S)$ is bounded then $\mathcal{V}(S \setminus \{s\}) \cap VR(s, S)$ is a tree.*

Let Γ be a closed Jordan curve in the plane large enough to enclose all the intersections of bisectors in $\mathcal{J}$, and such that each bisector crosses Γ exactly twice and transversally. Without loss of generality, we restrict all computations within Γ.[3] The curve Γ can be interpreted as $J(p, s_\infty)$, for all $p \in S$, where s_∞ is an additional site at infinity. Let the interior of Γ be denoted as D_Γ. Our *domain of computation* is $D_s = \mathrm{VR}(s, S) \cap D_\Gamma$, see Figure 3; we compute $\mathcal{V}(S \setminus \{s\}) \cap D_s$.

The following lemmas are used as tools in our proofs. Let C_p be a cycle of p-related bisectors in the arrangement of bisectors $\mathcal{J} \cup \Gamma$. If for every edge in C_p the label p appears on the outside of the cycle then C_p is called *p-inverse*, see Figure 4(a). If the label p appears only inside C_p then C_p is called a *p-cycle*, see Figure 4(b). By definition, $\mathrm{VR}(p, S) \subseteq C_p$ for any p-cycle C_p. A p-inverse cycle cannot contain pieces of Γ.

▶ **Lemma 2.** *In an admissible bisector system there is no p-inverse cycle.*

Proof. The farthest Voronoi region of p is $\mathrm{FVR}(p, S) = \bigcap_{q \in S \setminus \{p\}} D(q, p)$. By its definition, $\mathrm{FVR}(p, S)$ must be enclosed in any p-inverse cycle C_p. But farthest Voronoi regions must be unbounded [15, 3] deriving a contradiction. ◀

The following *transitivity lemma* is a consequence of transitivity of dominance regions [3, Lemma 2] and the fact that bisectors $J(p, q), J(q, r), J(p, r)$ intersect at the same point(s).

▶ **Lemma 3.** *Let $z \in \mathbb{R}^2$ and $p, q, r \in S$. If $z \in D(p, q)$ and $z \in \overline{D(q, r)}$, then $z \in D(p, r)$.*

We make a general position assumption that no three p-related bisectors intersect at the same point. This implies that Voronoi vertices have degree 3.

[3] The presence of Γ is conceptual and its exact position unknown; we never compute coordinates on Γ.

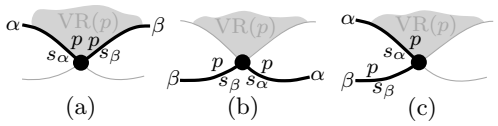


Figure 5 (a) Arcs α, β fulfill the p-monotone path condition; they do not fulfill it (b) and (c).

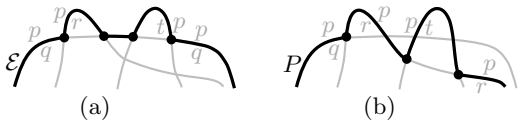


Figure 6 (a) The envelope $\mathcal{E} = \text{env}(\mathcal{J}_{p,\{q,r,t\}})$. (b) A p-monotone path P in $\mathcal{J}_{p,\{q,r,t\}}$.

3 Problem formulation and definitions

Let $\mathcal{S}$ denote the sequence of Voronoi edges along $\partial\text{VR}(s, S)$, i.e., $\mathcal{S} = \partial\text{VR}(s, S) \cap D_\Gamma$. We consider $\mathcal{S}$ as a cyclically ordered set of *arcs*, where each arc is a Voronoi edge of $\partial\text{VR}(s, S)$. Each arc $\alpha \in \mathcal{S}$ is induced by a site $s_\alpha \in S \setminus \{s\}$ such that $\alpha \subseteq J(s, s_\alpha)$. A site p may induce several arcs on $\mathcal{S}$; recall, that the sequence of site occurrences along $\partial\text{VR}(s, S)$ is a Davenport-Schinzel sequence of order 2.

We can interpret the arcs in $\mathcal{S}$ as sites that induce a Voronoi diagram $\mathcal{V}(\mathcal{S})$, where $\mathcal{V}(\mathcal{S}) = \mathcal{V}(S \setminus \{s\}) \cap D_s$ and $D_s = \text{VR}(s, S) \cap D_\Gamma$. Figure 7(a) illustrates $\mathcal{S}$ and $\mathcal{V}(\mathcal{S})$ in black (bold) and red, respectively. By Lemma 1, each face of $\mathcal{V}(S \setminus \{s\}) \cap D_s$ is incident to exactly one arc in $\mathcal{S}$. In this respect, each arc α in $\mathcal{S}$ has a Voronoi region, $\text{VR}(\alpha, \mathcal{S})$, which is the face of $\mathcal{V}(S \setminus \{s\}) \cap D_s$ incident to α.

For a site $p \in S$ and $S' \subseteq S$, let $\mathcal{J}_{p,S'} = \{J(p, q) \,|\, q \in S', q \neq p\}$ denote the set of all p-related bisectors involving sites in S'. The arrangement of a bisector set J is denoted by $\mathcal{A}(J)$. $\mathcal{A}(\mathcal{J}_{p,S'})$ may consist of more than one connected components.

▶ **Definition 4.** A *path* P in $\mathcal{J}_{p,S'}$ is a connected sequence of alternating edges and vertices of the arrangement $\mathcal{A}(\mathcal{J}_{p,S'})$. An *arc* α of P is a maximally connected set of consecutive edges and vertices of the arrangement along P, which belong to the same bisector. The common endpoint of two consecutive arcs of P is a *vertex of* P. An arc of P is also called an *edge*.

Two consecutive arcs in a path P are pieces of different bisectors. We use the notation $\alpha \in P$ for referring to an arc α of P. For $\alpha \in P$, let $s_\alpha \in S$ denote the site in S that *induces* α, where $\alpha \subseteq J(p, s_\alpha)$.

▶ **Definition 5.** A path P in $\mathcal{J}_{p,S'}$ is called p-*monotone* if any two consecutive arcs $\alpha, \beta \in P$, where $\alpha \subseteq J(p, s_\alpha)$ and $\beta \subseteq J(p, s_\beta)$, induce the Voronoi edges of $\partial\text{VR}(p, \{p, s_\alpha, s_\beta\})$, which are incident to the common endpoint of α, β (see Figure 5).

▶ **Definition 6.** The *envelope of* $\mathcal{J}_{p,S'}$, with respect to site p, is $\text{env}(\mathcal{J}_{p,S'}) = \partial\text{VR}(p, S' \cup \{p\})$, called a p-envelope (see Figure 6(a)).

Figure 6 illustrates two p-monotone paths, where the path in Figure 6(a) is a p-envelope. Notice, $\mathcal{S}$ is the envelope of the s-related bisectors in $\mathcal{J}$, $\mathcal{S} = \text{env}(\mathcal{J}_{s,S\setminus\{s\}}) \cap D_\Gamma$. A p-monotone path that is not a p-envelope can be a Davenport-Schinzel sequence of order > 2, with respect to site occurrences in $S \setminus \{s\}$.

The system of bisectors $\mathcal{J}_{p,S'}$ may consist of several connected components. For convenience, in order to unify the various connected components of $\mathcal{A}(\mathcal{J}_{p,S'})$ and to consider its p-monotone paths as single curves, we include the curve Γ in the corresponding system of bisectors. Then, $\text{env}(\mathcal{J}_{p,S'} \cup \Gamma)$ is a closed p-monotone path, whose connected components in $\mathcal{J}_{p,S'}$ are interleaved with arcs of Γ.

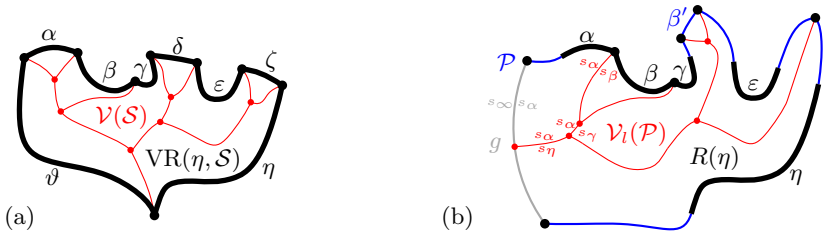


Figure 7 (a) illustrates $\mathcal{S}$ in black (bold) and $\mathcal{V}(\mathcal{S})$ in red, $\mathcal{S} = (\alpha, \beta, \gamma, \delta, \varepsilon, \zeta, \eta, \vartheta)$. (b) illustrates $\mathcal{V}_l(\mathcal{P})$ for a boundary curve $\mathcal{P} = (\alpha, \beta, \gamma, \beta', \varepsilon, \eta, g)$ for $\mathcal{S}'$, where $\mathcal{S}' = (\alpha, \beta, \gamma, \varepsilon, \eta)$ is shown in bold. The arcs of $\mathcal{P}$ are original except the auxiliary arc β' and the Γ-arc g.

▶ **Definition 7.** Consider $\mathcal{S}' \subseteq \mathcal{S}$ and let $S' = \{s_\alpha \in S \mid \alpha \in \mathcal{S}'\} \subseteq S \backslash \{s\}$ be its corresponding set of sites. A closed s-monotone path in $\mathcal{J}_{s,S'} \cup \Gamma$ that contains all arcs in $\mathcal{S}'$ is called a *boundary curve for* $\mathcal{S}'$. The part of the plane enclosed in a boundary curve $\mathcal{P}$ is called the *domain* of $\mathcal{P}$, and it is denoted by $D_\mathcal{P}$. Given $\mathcal{P}$, we also use notation $S_\mathcal{P}$ to denote S'.

A set of arcs $\mathcal{S}' \subset \mathcal{S}$ can admit several different boundary curves. One such boundary curve is its envelope $\mathcal{E} = \mathrm{env}(\mathcal{J}_{s,S'} \cup \Gamma)$. Figure 7(b) illustrates a boundary curve for $\mathcal{S}' \subseteq \mathcal{S}$, where $\mathcal{S}$ is the set of arcs in Figure 7(a).

A boundary curve $\mathcal{P}$ in $\mathcal{J}_{s,S'} \cup \Gamma$ consists of pieces of bisectors in $\mathcal{J}_{s,S'}$, called *boundary arcs*, and pieces of Γ, called Γ-*arcs*. Γ-arcs 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 others are called *auxiliary arcs*. Original boundary arcs are expanded versions of the arcs in $\mathcal{S}'$. To distinguish between them, we call the elements of $\mathcal{S}$ *core arcs* and use an * in their notation. In Figure 7 the core arcs are illustrated in bold.

For a set of arcs $\mathcal{S}' \subseteq \mathcal{S}$, we define the Voronoi diagram of $\mathcal{S}' \subseteq \mathcal{S}$ as $\mathcal{V}(\mathcal{S}') = \mathcal{V}(S') \cap D_\mathcal{E}$, where $\mathcal{E}$ is the s-envelope $\mathrm{env}(\mathcal{J}_{s,S'} \cup \Gamma)$. $\mathcal{V}(\mathcal{S}')$ can be regarded as the Voronoi diagram of the envelope $\mathcal{E}$, thus, it can also be denoted as $\mathcal{V}(\mathcal{E})$. The face of $\mathcal{V}(\mathcal{S}')$ incident to an arc $\alpha \in \mathcal{E}$ is called the Voronoi region of α and is denoted by $\mathrm{VR}(\alpha, \mathcal{S}')$. We would like to extend the definition of $\mathcal{V}(\mathcal{S}')$ to any boundary curve stemming out of $\mathcal{S}'$. To this goal we define a *Voronoi-like diagram* for any boundary curve $\mathcal{P}$ of $\mathcal{S}'$. Notice, $D_s \subseteq D_\mathcal{E} \subseteq D_\mathcal{P}$.

▶ **Definition 8.** Given a boundary curve $\mathcal{P}$ in $\mathcal{J}_{s,S'} \cup \Gamma$, a *Voronoi-like diagram* of $\mathcal{P}$ is a plane graph on $\mathcal{J}(S') = \{J(p,q) \in \mathcal{J} \mid p, q \in S'\}$ inducing a subdivision on the domain $D_\mathcal{P}$ as follows (see Figure 7(b)):

1. There is exactly one face $R(\alpha)$ for each boundary arc α of $\mathcal{P}$, and $\partial R(\alpha)$ consists of the arc α plus an s_α-monotone path in $\mathcal{J}_{s_\alpha,S'} \cup \Gamma$.
2. $\bigcup_{\alpha \in \mathcal{P} \backslash \Gamma} \overline{R(\alpha)} = \overline{D_\mathcal{P}}$.

The Voronoi-like diagram of $\mathcal{P}$ is $\mathcal{V}_l(\mathcal{P}) = D_\mathcal{P} \setminus \bigcup_{\alpha \in \mathcal{P}} R(\alpha)$.

Voronoi-like regions in $\mathcal{V}_l(\mathcal{P})$ are related to real Voronoi regions in $\mathcal{V}(\mathcal{S}')$ as supersets, as shown in the following lemma. In Figure 7(b) the Voronoi-like region $R(\eta)$ is a superset of its corresponding Voronoi region $\mathrm{VR}(\eta, \mathcal{S})$ in (a); similarly for e.g., $R(\alpha)$. Note that not every boundary curve of $\mathcal{S}' \subset \mathcal{S}$ needs to admit a Voronoi-like diagram.

▶ **Lemma 9.** *Let α be a boundary arc in a boundary curve $\mathcal{P}$ of $\mathcal{S}'$ such that a portion $\tilde{\alpha} \subseteq \alpha$ appears on the s-envelope $\mathcal{E}$ of $\mathcal{S}'$, $\mathcal{E} = env(\mathcal{J}_{s,S'} \cup \Gamma)$. Given $\mathcal{V}_l(\mathcal{P})$, $R(\alpha) \supseteq VR(\tilde{\alpha}, \mathcal{S}')$. If α is original, then $R(\alpha) \supseteq VR(\tilde{\alpha}, \mathcal{S}') \supseteq VR(\alpha^*, \mathcal{S})$.*

Proof. By the definition of a Voronoi region, no piece of a bisector $J(s_\alpha, \cdot)$ can appear in the interior of $\mathrm{VR}(\tilde{\alpha}, \mathcal{S}')$, where $\tilde{\alpha} \in \mathcal{E}$ (recall that $\mathcal{V}(\mathcal{S}') = \mathcal{V}(\mathcal{E})$). Since in addition $\alpha \supseteq \tilde{\alpha}$, the

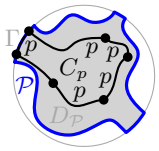


Figure 8 A p-cycle (possibly with Γ-arcs) within $\overline{D_{\mathcal{P}}}$ does not exist.

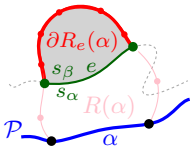


Figure 9 The shaded region lies in $D(s_\beta, s_\alpha)$.

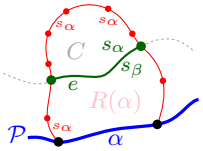


Figure 10 Edge e has a contradictory edge labeling.

claim follows. For an original arc α, since $S' \subseteq S$, by the monotonicity property of Voronoi regions, we also have $\mathrm{VR}(\tilde{\alpha}, \mathcal{S}') \supseteq \mathrm{VR}(\alpha^*, \mathcal{S})$. ◀

As a corollary to Lemma 9, the adjacencies of the real Voronoi diagram $\mathcal{V}(\mathcal{S}')$ are preserved in $\mathcal{V}_l(\mathcal{P})$, for all arcs that are common to the envelope $\mathcal{E}$ and the boundary curve $\mathcal{P}$. In addition, $\mathcal{V}_l(\mathcal{E})$ coincides with the real Voronoi diagram $\mathcal{V}(\mathcal{S}')$.

▶ **Corollary 10.** *$\mathcal{V}_l(\mathcal{E}) = \mathcal{V}(\mathcal{S}')$. This also implies $\mathcal{V}_l(\mathcal{S}) = \mathcal{V}(\mathcal{S})$.*

The following Lemma 12 gives a basic property of Voronoi-like regions that is essential for subsequent proofs. To establish it we first need the following observation.

▶ **Lemma 11.** *$\overline{D_{\mathcal{P}}}$ cannot contain a p-cycle of $\mathcal{J}(S_{\mathcal{P}}) \cup \Gamma$, for any $p \in S_{\mathcal{P}}$.*

Proof. Let $p \in S_{\mathcal{P}}$ define an original arc along $\mathcal{P}$. This arc is bounding $\mathrm{VR}(p, S_{\mathcal{P}} \cup \{s\})$, thus, it must have a portion within $\mathrm{VR}(p, S_{\mathcal{P}})$. Hence, $\mathrm{VR}(p, S_{\mathcal{P}})$ has a non-empty intersection with $\mathbb{R}^2 \setminus \overline{D_{\mathcal{P}}}$. But $\mathrm{VR}(p, S_{\mathcal{P}})$ must be enclosed within any p-cycle of $\mathcal{J}(S_{\mathcal{P}}) \cup \Gamma$, by its definition. Thus, no such p-cycle can be contained in $\overline{D_{\mathcal{P}}}$. Refer to Figure 8. ◀

▶ **Lemma 12.** *Suppose bisector $J(s_\alpha, s_\beta)$ appears within $R(\alpha)$ (see Figure 9). For any connected component e of $J(s_\alpha, s_\beta) \cap R(\alpha)$ that is not intersecting α, the label s_α must appear on the same side of e as α. Let $\partial R_e(\alpha)$ denote the portion of $\partial R(\alpha)$* cut out *by such a component e, at opposite side from α. Then $\partial R_e(\alpha) \subseteq D(s_\beta, s_\alpha)$.*

By Lemma 12, any components of $J(s_\alpha, s_\beta) \cap R(\alpha)$ must appear sequentially along $\partial R(\alpha)$. Note that $\partial R_e(\alpha)$ may as well contain Γ-arcs.

Proof. Suppose for the sake of contradiction that there is such a component $e \subseteq J(s_\alpha, s_\beta) \cap R(\alpha)$ with the label s_α appearing at opposite side of e as α (see Figure 10). Then e and $\partial R(\alpha)$ form an s_α-cycle C within $\overline{D_{\mathcal{P}}}$, contradicting Lemma 11. Suppose now that $\partial R_e(\alpha)$ lies only partially in $D(s_\beta, s_\alpha)$. Then $J(s_\beta, s_\alpha)$ would have to re-enter $R(\alpha)$ at $\partial R_e(\alpha)$, resulting in another component of $J(s_\beta, s_\alpha) \cap R(\alpha)$ with an invalid labeling. ◀

The following lemma extends Lemma 12 when a component e of $J(s_\alpha, s_\beta) \cap R(\alpha)$ intersects arc α. If $J(s_\alpha, s_\beta)$ intersects α, then there is also a component $\tilde{\beta}$ of $J(s, s_\beta) \cap R(\alpha)$ intersecting α at the same point as e. If $\tilde{\beta}$ has only one endpoint on α, let $\partial R_e(\alpha)$ denote the portion of $\partial R(\alpha)$ that is *cut out* by e, at the side of its s_β-label (see Figure 11(a)). If both endpoints of $\tilde{\beta}$ are on α then there are two components of $J(s_\alpha, s_\beta) \cap R(\alpha)$ incident to α (see Figure 11(b)); let $\partial R_e(\alpha)$ denote the portion of $\partial R(\alpha)$ between these two components.

▶ **Lemma 13.** *Let e be a component of $J(s_\alpha, s_\beta) \cap R(\alpha)$. Then $\partial R_e(\alpha) \subseteq D(s_\beta, s_\alpha)$.*

Using the basic property of Lemma 12 and its extension, we show that if there is any non-empty component of $J(s_\alpha, s_\beta) \cap R(\alpha)$, then $J(s, s_\beta)$ must also intersect $D_{\mathcal{P}}$, i.e., there exists a non-empty component of $J(s, s_\beta) \cap D_{\mathcal{P}}$ that is *missing from* $\mathcal{P}$. Using this property and Theorem 18 of the next section, we obtain the following theorem (see Section 5).

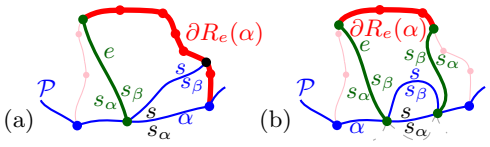


Figure 11 Illustrations for Lemma 13. The bold red parts $\partial R_e(\alpha)$ belong to $D(s_\beta, s_\alpha)$.

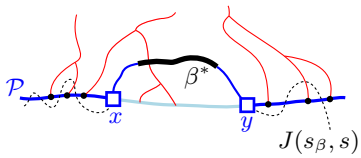


Figure 12 $\mathcal{P}_\beta = \mathcal{P} \oplus \beta$, core arc β^* is bold, black. Endpoints of β are x, y.

▶ **Theorem 14.** *Given a boundary curve $\mathcal{P}$ of $\mathcal{S}' \subseteq \mathcal{S}$, $\mathcal{V}_l(\mathcal{P})$ (if it exists) is unique.*

The complexity of $\mathcal{V}_l(\mathcal{P})$ is $O(|\mathcal{P}|)$, where $|\mathcal{P}|$ denotes the number of boundary arcs in $\mathcal{P}$, as it is a planar graph with exactly one face per boundary arc and vertices of degree 3 (or 1).

4 Insertion in a Voronoi-like diagram

Consider a boundary curve $\mathcal{P}$ for $\mathcal{S}' \subset \mathcal{S}$ and its Voronoi-like diagram $\mathcal{V}_l(\mathcal{P})$. Let β^* be an arc in $\mathcal{S} \setminus \mathcal{S}'$, thus, β^* is contained in the closure of the domain $\overline{D_\mathcal{P}}$.

We define arc $\beta \supseteq \beta^*$ as the connected component of $J(s, s_\beta) \cap \overline{D_\mathcal{P}}$ that contains β^* (see Figure 12). We also define an insertion operation $\oplus$, which inserts arc β in $\mathcal{P}$ deriving a new boundary curve $\mathcal{P}_\beta = \mathcal{P} \oplus \beta$, and also inserts $R(\beta)$ in $\mathcal{V}_l(\mathcal{P})$ deriving the Voronoi-like diagram $\mathcal{V}_l(\mathcal{P}_\beta) = \mathcal{V}_l(\mathcal{P}) \oplus \beta$. $\mathcal{P}_\beta$ is the boundary curve obtained by deleting the portion of $\mathcal{P}$ between the endpoints of β, which lies in $D(s_\beta, s)$, and substituting it with β.

Figure 13 enumerates the possible cases of inserting arc β in $\mathcal{P}$ and is summarized in the following observation.

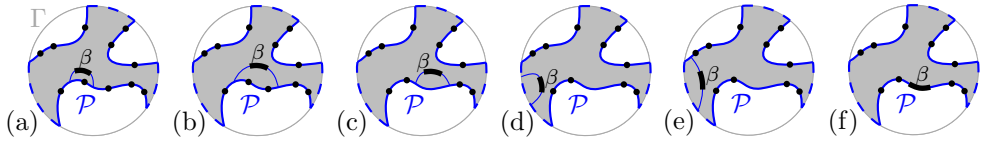


Figure 13 Insertion cases for an arc β.

▶ **Observation 15.** *Possible cases of inserting arc β in $\mathcal{P}$ (see Figure 13). $D_{\mathcal{P}_\beta} \subseteq D_\mathcal{P}$.*

(a) *β straddles the endpoint of two consecutive boundary arcs; no arcs in $\mathcal{P}$ are deleted.*
(b) *Auxiliary arcs in $\mathcal{P}$ are deleted by β; their regions are also deleted from $\mathcal{V}_l(\mathcal{P}_\beta)$.*
(c) *An arc $\alpha \in \mathcal{P}$ is split into two arcs by β; $R(\alpha)$ in $\mathcal{V}_l(\mathcal{P})$ will also be split.*
(d) *A Γ-arc is split in two by β; $\mathcal{V}_l(\mathcal{P}_\beta)$ may switch from being a tree to being a forest.*
(e) *A Γ-arc is deleted or shrunk by inserting β. $\mathcal{V}_l(\mathcal{P}_\beta)$ may become a tree.*
(f) *$\mathcal{P}$ already contains a boundary arc $\bar{\beta} \supseteq \beta^*$; then $\beta = \bar{\beta}$ and $\mathcal{P}_\beta = \mathcal{P}$.*

Note that $\mathcal{P}_\beta$ may contain fewer, the same number, or even one extra auxiliary arc compared to $\mathcal{P}$.

▶ **Lemma 16.** *The curve $\mathcal{P}_\beta = \mathcal{P} \oplus \beta$ is a boundary curve for $\mathcal{S}' \cup \{\beta^*\}$.*

Proof. Since $\mathcal{P}$ is a (closed) s-monotone path in $\mathcal{J}_{s,\mathcal{S}'} \cup \Gamma$, $\mathcal{P}_\beta$ is also such a path in $\mathcal{J}_{s,\mathcal{S}' \cup \{s_\beta\}} \cup \Gamma$, by construction. No original arc in $\mathcal{P}$ can be deleted by the insertion of β, because every core arc in $\mathcal{S}$ appears on the envelope $\text{env}(\mathcal{J}_{s,\mathcal{S}} \cup \Gamma)$; thus, such an arc cannot be *cut out* by the insertion of β on $\mathcal{P}$. Hence, $\mathcal{P}_\beta$ contains all arcs in $\mathcal{S}' \cup \{\beta^*\}$. ◀

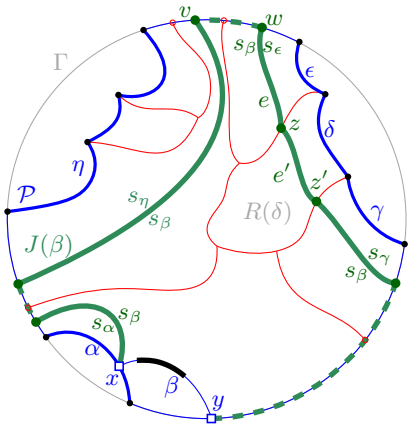


Figure 14 The merge curve $J(\beta)$ (thick, green) on $\mathcal{V}_l(\mathcal{P})$ (thin, red).

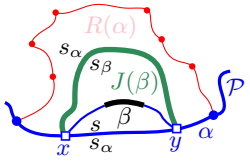


Figure 15 If β splits α, $J(\beta) \subset R(\alpha)$ would yield a forbidden s_α-inverse cycle.

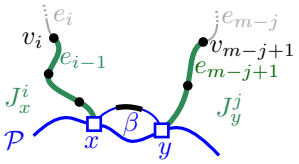


Figure 16 J^i_x and J^j_y in Section 4.1.

Given $\mathcal{V}_l(\mathcal{P})$ and arc β, where $\beta^* \in \mathcal{S} \setminus \mathcal{S}'$, we define a *merge curve* $J(\beta)$, within $\mathcal{V}_l(\mathcal{P})$, which delimits the boundary of $R(\beta)$ in $\mathcal{V}_l(\mathcal{P}_\beta)$. We define $J(\beta)$ incrementally, starting at an endpoint of β. Let x and y denote the endpoints of β, where x, β, y are in counterclockwise order around $\mathcal{P}_\beta$; refer to Figure 14.

▶ **Definition 17.** Given $\mathcal{V}_l(\mathcal{P})$ and arc $\beta \subset J(s, s_\beta)$, the *merge curve* $J(\beta)$ is a path $(v_1, \ldots, v_m)$ in the arrangement of s_β-related bisectors, $\mathcal{J}_{s_\beta, S_\mathcal{P}} \cup \Gamma$, connecting the endpoints of β, $v_1 = x$ and $v_m = y$. Each edge $e_i = (v_i, v_{i+1})$ is an arc of a bisector $J(s_\beta, \cdot)$, called an *ordinary* edge, or an arc on Γ. For $i = 1$: if $x \in J(s_\beta, s_\alpha)$, then $e_1 \subseteq J(s_\beta, s_\alpha)$; if $x \in \Gamma$, then $e_1 \subseteq \Gamma$. Given v_i, vertex v_{i+1} and edge e_{i+1} are defined as follows (see Figure 14). Wlog we assume a clockwise ordering of $J(\beta)$.

1. If $e_i \subseteq J(s_\beta, s_\alpha)$, let v_{i+1} be the other endpoint of the component $J(s_\beta, s_\alpha) \cap R(\alpha)$ incident to v_i. If $v_{i+1} \in J(s_\beta, \cdot) \cap J(s_\beta, s_\alpha)$, then $e_{i+1} \subseteq J(s_\beta, \cdot)$. If $v_{i+1} \in \Gamma$, then $e_{i+1} \subseteq \Gamma$. (In Figure 14, see $e_i = e', v_i = z, v_{i+1} = z'$.)
2. If $e_i \subseteq \Gamma$, let g be the Γ-arc incident to v_i. Let $e_{i+1} \subseteq J(s_\beta, s_\gamma)$, where $R(\gamma)$ is the first region, incident to g clockwise from v_i, such that $J(s_\beta, s_\gamma)$ intersects $g \cap \overline{R(\gamma)}$; let v_{i+1} be this intersection point. (In Figure 14, see $v_i = v$ and $v_{i+1} = w$.)

A vertex v along $J(\beta)$, is called *valid* if v is a vertex in the arrangement $\mathcal{A}(\mathcal{J}_{s_\beta, S_\mathcal{P}} \cup \Gamma)$ or v is an endpoint of β. The following theorem shows that $J(\beta)$ is well defined, given $\mathcal{V}_l(\mathcal{P})$, and that it forms an s_β-monotone path. We defer its proof to the end of this section.

▶ **Theorem 18.** *$J(\beta)$ is a unique s_β-monotone path in the arrangement of s_β-related bisectors $\mathcal{J}_{s_\beta, S_\mathcal{P}} \cup \Gamma$ connecting the endpoints of β. $J(\beta)$ can contain at most one ordinary edge per region of $\mathcal{V}_l(\mathcal{P})$, with the exception of e_1 and e_{m-1}, when v_1 and v_m are incident to the same face in $\mathcal{V}_l(\mathcal{P})$. $J(\beta)$ cannot intersect the interior of arc β.*

We define $R(\beta)$ as the area enclosed by $\beta \cup J(\beta)$. Let $\mathcal{V}_l(\mathcal{P}) \oplus \beta$ be the subdivision of $D_{\mathcal{P}_\beta}$ obtained by inserting $J(\beta)$ in $\mathcal{V}_l(\mathcal{P})$ and deleting any portion of $\mathcal{V}_l(\mathcal{P})$ enclosed by $J(\beta)$, i.e., $\mathcal{V}_l(\mathcal{P}) \oplus \beta = \big((\mathcal{V}_l(\mathcal{P}) \setminus R(\beta)) \cup J(\beta)\big) \cap D_{\mathcal{P}_\beta}$. We prove that $\mathcal{V}_l(\mathcal{P}) \oplus \beta$ is a Voronoi-like diagram. To this goal we need an additional property of $J(\beta)$.

▶ **Lemma 19.** *If the insertion of β splits an arc $\alpha \in \mathcal{P}$ (Observation 15(c)), then $J(\beta)$ also splits $R(\alpha)$ and $J(\beta) \not\subseteq R(\alpha)$. In no other case can $J(\beta)$ split a region $R(\alpha)$ in $\mathcal{V}_l(\mathcal{P})$.*

Proof. Suppose for the sake of contradiction that β splits arc α and $J(\beta) \subset R(\alpha)$, as shown in Figure 15. Then $J(\beta) = J(s_\alpha, s_\beta) \cap R(\alpha)$ and the bisector $J(s_\alpha, s_\beta)$ together with the arc α form a forbidden s_α-inverse cycle, deriving a contradiction to Lemma 2. Thus, $J(\beta)$ must intersect $\partial R(\alpha)$ in $\mathcal{V}_l(\mathcal{P})$ and therefore $J(\beta) \not\subseteq R(\alpha)$. By Theorem 18, $J(\beta)$ can only enter some other region at most once. Thus, $J(\beta)$ cannot split any other region. ◀

▶ **Theorem 20.** *$\mathcal{V}_l(\mathcal{P}) \oplus \beta$ is a Voronoi-like diagram for $\mathcal{P}_\beta = \mathcal{P} \oplus \beta$, denoted $\mathcal{V}_l(\mathcal{P}_\beta)$.*

Proof. By Theorem 18, $R(\beta)$ fulfills the properties of a Voronoi-like region. Moreover, the updated boundary of any other region $R(\alpha)$ in $\mathcal{V}_l(\mathcal{P})$, which is truncated by $J(\beta)$, remains an s_α-monotone path. By Lemma 19, $J(\beta)$ cannot split a region $R(\alpha)$ in $\mathcal{V}_l(\mathcal{P})$, and thus, it cannot create a face that is not incident to α. Therefore, $\mathcal{V}_l(\mathcal{P}) \oplus \beta$ fulfills all properties of Definition 8. ◀

The tracing of $J(\beta)$ within $\mathcal{V}_l(\mathcal{P})$, given the endpoints of β, can be done similarly to any ordinary Voronoi diagram, see e.g., [11] [2, Ch. 7.5.3] for AVDs, or [9, Ch. 7.4] [18, Ch. 5.5.2.1] for concrete diagrams. For a Voronoi-like diagram this can be established due to the basic property of Lemmas 12 and 13.

Special care is required in cases (c), (d), and (e) of Observation 15, in order to identify the first edge of $J(\beta)$; in these cases, β may not overlap with any feature of $\mathcal{V}_l(\mathcal{P})$, thus, a starting point for tracing $J(\beta)$ is not readily available. In case (c), we trace a portion of $\partial R(\alpha)$, which does not get deleted afterwards, thus it adds to the time complexity of the operation $\mathcal{V}_l(\mathcal{P}) \oplus \beta$ (see Lemma 21). In cases (d) and (e), we show that if no feature of $\mathcal{V}_l(\mathcal{P})$ overlaps β, then either there is a leaf of $\mathcal{V}_l(\mathcal{P})$ in the neighboring Γ-arc or $J(\beta) \subseteq \overline{R(\alpha)}$. In either case a starting point for $J(\beta)$ can be identified in $O(1)$ time. Notice, if $J(\beta) \subseteq \overline{R(\alpha)}$, then it consists of a single bisector $J(s_\beta, s_\alpha)$ (and one or two Γ-arcs).

The following lemma gives the time complexity to compute $J(\beta)$ and update $\mathcal{V}_l(\mathcal{P}_\beta)$. The statement of the lemma is an adaptation from [10], however, the proof contains cases that do not appear in a farthest segment Voronoi diagram. $|\cdot|$ denotes complexity.

Let $\tilde{\mathcal{P}}$ denote a finer version of $\mathcal{P}$, where a Γ-arc between two consecutive boundary arcs in $\mathcal{P}$ is partitioned into smaller Γ-arcs as defined by the incident faces of $\mathcal{V}_l(\mathcal{P})$. Since $|\mathcal{V}_l(\mathcal{P})|$ is $O(|\mathcal{P}|)$, $|\tilde{\mathcal{P}}|$ is also $O(|\mathcal{P}|)$.

▶ **Lemma 21.** *Let α and γ be the first original arcs on $\mathcal{P}_\beta$ occurring before and after β. Let $d(\beta)$ be the number of arcs in $\tilde{\mathcal{P}}$ between α and γ (both boundary and Γ-arcs). Given α, γ, and $\mathcal{V}_l(\mathcal{P})$, in all cases of Observation 15, except (c), the merge curve $J(\beta)$ and the diagram $\mathcal{V}_l(\mathcal{P}_\beta)$ can be computed in time $O(|R(\beta)| + d(\beta))$. In case (c), where an arc is split and a new arc ω is created by the insertion of β, the time is $O(|\partial R(\beta)| + |\partial R(\omega)| + d(\beta))$.*

4.1 Proving Theorem 18

We first establish that $J(\beta)$ cannot intersect arc β, other than its endpoints, using the following Lemma.

▶ **Lemma 22.** *Given $\mathcal{V}_l(\mathcal{P})$, for any arc $\alpha \in \mathcal{P}$, $R(\alpha) \subseteq D(s, s_\alpha)$.*

Proof. The contrary would yield an s_α-inverse cycle defined by $J(s, s_\alpha)$ and $\partial R(\alpha)$. ◀

Lemma 22 implies that bisector $J(s_\beta, s_\alpha)$ cannot intersect $J(s, s_\beta)$ within region $R(\alpha)$. Thus $J(\beta)$ cannot intersect arc β in its interior. The following lemma is used in several proofs.

▶ **Lemma 23.** *$D(s, \cdot) \cap D_{\mathcal{P}}$ is always connected. Thus, any components of $J(s, \cdot) \cap D_{\mathcal{P}}$ must appear sequentially along $\mathcal{P}$.*

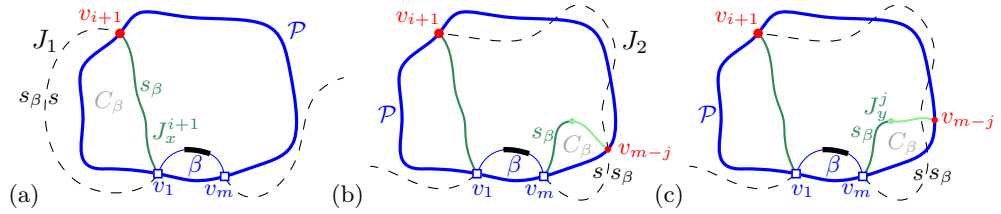


Figure 17 The assumption that edge $e_i = (v_i, v_{i+1})$ of the merge curve J_x^i hits a boundary arc of $\mathcal{P}$ as in Lemma 24.

Proof. If we assume the contrary we obtain an s-inverse cycle defined by $J(s,\cdot)$ and $\mathcal{P}$. ◀

To prove Theorem 18 we use a bi-directional induction on the vertices of $J(\beta)$. Let $J_x^i = (v_1, v_2, \ldots, v_i), 1 \le i < m$, be the subpath of $J(\beta)$ starting at $v_1 = x$ up to vertex v_i, including a small neighborhood of e_i incident to v_i, see Figure 16. Note that vertex v_i uniquely determines e_i, however, its other endpoint is not yet specified. Similarly, let $J_y^j = (v_m, v_{m-1}, \ldots, v_{m-j+1}), 1 \le j < m$, denote the subpath of $J(\beta)$, starting at v_m up to vertex v_{m-j+1}, including a small neighborhood of edge e_{m-j}. Recall that we refer to the edges of $J(\beta)$ that are not Γ-arcs as *ordinary*. For any ordinary edge $e_\ell \in J(\beta)$, let α_ℓ denote the boundary arc that *induces* e_ℓ, i.e., $e_\ell \subseteq J(s_{\alpha_\ell}, s_\beta) \cap R(\alpha_\ell)$.

Induction hypothesis: Suppose J_x^i and J_y^j, $i, j \ge 1$, are disjoint s_β-monotone paths. Suppose further that each ordinary edge of J_x^i and of J_y^j passes through a distinct region of $\mathcal{V}_l(\mathcal{P})$: α_ℓ is distinct for ℓ, $1 \le \ell \le i$ and $m - j \le \ell < m$, except possibly $\alpha_i = \alpha_{m-j}$ and $\alpha_1 = \alpha_{m-1}$.

Induction step: Assuming that $i + j < m$, we prove that at least one of J_x^i or J_y^j can respectively grow to J_x^{i+1} or J_y^{j+1} at a valid vertex (Lemmas 24, 25), and it enters a new region of $\mathcal{V}_l(\mathcal{P})$ that has not been visited so far (Lemma 27). A finish condition when $i + j = m$ is given in Lemma 26. The base case for $i = j = 1$ is trivially true.

Suppose that $e_i \subseteq J(s_{\alpha_i}, s_\beta)$ and $v_i \in \partial R(\alpha_i)$. To show that v_{i+1} is a valid vertex it is enough to show that (1) v_{i+1} can not be on α_i, and (2) if v_i is on a Γ-arc then v_{i+1} can be determined on the same Γ-arc. However, we cannot easily derive these conclusions directly. Instead we show that if v_{i+1} is not valid then v_{m-j} will have to be valid.

In the following lemmas we assume that the induction hypothesis holds.

▶ **Lemma 24.** *Suppose $e_i \subseteq J(s_{\alpha_i}, s_\beta)$ but $v_{i+1} \in \alpha_i$, i.e., it is not a valid vertex because e_i hits α_i. Then vertex v_{m-j} must be a valid vertex in $\mathcal{A}(\mathcal{J}_{s_\beta, S_\mathcal{P}})$, and v_{m-j} can not be on $\mathcal{P}$.*

▶ **Lemma 25.** *Suppose vertex v_i is on a Γ-arc g but v_{i+1} cannot be determined because no bisector $J(s_\beta, s_\gamma)$ intersects $\overline{R(\gamma)} \cap g$, clockwise from v_i. Then vertex v_{m-j} must be a valid vertex in $\mathcal{A}(\mathcal{J}_{s_\beta, S_\mathcal{P}})$ and v_{m-j} can not be on $\mathcal{P}$.*

Proof of Lemma 24. Suppose vertex v_{i+1} of e_i lies on α_i as shown in Figure 17(a). Vertex v_{i+1} is the intersection point of related bisectors $J(s, s_{\alpha_i})$, $J(s_\beta, s_{\alpha_i})$ and thus also of $J(s, s_\beta)$. Observe that arc β partitions $J(s, s_\beta)$ in two parts: J_1 incident to v_1 and J_2 incident to v_m. We claim that v_{i+1} lies on J_2. Suppose otherwise, then J_x^{i+1} and J_1 would form a forbidden s_β-inverse cycle, see the dashed black and the green solid curve in Figure 17(a). By Lemma 23 the components of $J_2 \cap D_\mathcal{P}$ appear on $\mathcal{P}$ clockwise after v_{i+1} and before v_m, as shown in Figure 17(b) illustrating $J(s, s_\beta)$ as a black dashed curve.

Now consider J_y^j. We show that v_{m-j} cannot be on $\mathcal{P}$. First observe that v_{m-j} can not lie on $\mathcal{P}$, clockwise after v_m and before v_1, since J_y^{j+1} cannot cross β. We prove that v_{m-j}

cannot lie on $\mathcal{P}$ clockwise after v_1 and before v_{i+1}. To see that, note that edge e_{m-j} cannot cross any non-Γ edge of J_x^{i+1}, because α_{m-j} is distinct from all $\alpha_\ell, \ell \leq i$ (by the induction hypothesis). In addition, by the definition of a Γ-arc, v_{m-j} cannot lie on any Γ-arc of J_x^i. Finally, we show that v_{m-j} cannot lie on $\mathcal{P}$ clockwise after v_{i+1} and before v_m. If v_{m-j} lay on the boundary arc α_{m-j} then we would have $v_{m-j} \in J(s, s_\beta)$. This would define an s_β-inverse cycle C_β, formed by J_y^{j+1} and $J(s_\beta, s)$, see Figure 17(b). If v_{m-j} lay on a Γ-arc then there would also be a forbidden s_β-inverse cycle formed by J_y^{j+1} and $J(s, s_\beta)$ because in order to reach Γ edge e_i must cross $J(s, s_\beta)$. See the dashed black and the green curve in Figure 17(c). Thus $v_{m-j} \notin \mathcal{P}$.

Since $v_{m-j} \in \partial R(\alpha_{i+1})$ but $v_{m-j} \notin \mathcal{P}$, it must be a vertex of $\mathcal{A}(\mathcal{J}_{s_\beta, S_\mathcal{P}})$. ◀

Lemma 26 provides a finish condition for the induction. When it is met, $J(\beta) = J_x^i \cup J_y^j$, i.e., a concatenation of J_x^i and J_y^j.

▶ **Lemma 26.** *Suppose $i + j > 2$ and either (1) or (2) holds: (1) $\alpha_i = \alpha_{m-j}$, i.e., v_i and v_{m-j+1} are incident to a common region $R(\alpha_i)$ and $e_i, e_{m-j} \subseteq J(s_\beta, s_{\alpha_i})$; or (2) v_i and v_{m-j+1} are on a common Γ-arc g of $\mathcal{P}$ and $e_i, e_{m-j} \subseteq \Gamma$. Then $v_{i+1} = v_{m-j+1}$, $v_{m-j} = v_i$, and $m = i + j$.*

▶ **Lemma 27.** *Suppose vertex v_{i+1} is valid and $e_{i+1} \subseteq J(s_\beta, s_{a_{i+1}})$. Then $R(\alpha_{i+1})$ has not been visited by J_x^i nor J_y^j, i.e., $\alpha_{i+1} \neq \alpha_\ell$ for $\ell \leq i$ and for $m - j < \ell$.*

By Lemma 27, J_x^{i+1} and J_y^{j+1} always enter a new region of $\mathcal{V}_l(\mathcal{P})$ that has not been visited yet; thus, conditions (1) or (2) of Lemma 26 must be fulfilled at some point of the induction. Hence, the proof of Theorem 18 is complete. Completing the induction establishes also that the conditions of Lemmas 24 and 25 can never be met, thus, no vertex of $J(\beta)$ can be on a boundary arc of $\mathcal{P}$, except its endpoints.

5 $\mathcal{V}_l(\mathcal{P})$ is unique

In this section we establish that $\mathcal{V}_l(\mathcal{P})$ is unique. To this goal we prove the following lemma and use it to prove Theorem 14.

▶ **Lemma 28.** *Suppose there is a non-empty component e of $J(s_\alpha, \cdot)$ intersecting $R(\alpha)$ in $\mathcal{V}_l(\mathcal{P})$. Then $J(s, \cdot)$ must also intersect $D_\mathcal{P}$. Further, there exists a component of $J(s, \cdot) \cap D_\mathcal{P}$, denoted as β, such that the merge curve $J(\beta)$ in $\mathcal{V}_l(\mathcal{P})$ contains e.*

Proof sketch of Theorem 14. Suppose that for a given boundary curve $\mathcal{P}$ there exist two different Voronoi-like diagrams $\mathcal{V}_l^1 \neq \mathcal{V}_l^2$. Then there must be an edge $e^1 \subseteq J(s_\beta, s_\alpha)$ of $\mathcal{V}_l^1$, such that e^1 intersects region $R^2(\alpha)$ of $\mathcal{V}_l^2$. Let edge $e \subseteq J(s_\beta, s_\alpha)$ be the component of $R^2(\alpha) \cap J(s_\beta, s_\alpha)$ overlapping with e^1. Lemma 28 yields a non-empty component β_0 of $J(s, s_\beta) \cap D_\mathcal{P}$ such that $J(\beta_0)$ on $\mathcal{V}_l^2$ contains edge e. Since $J(\beta_0)$ and $\partial R^1(\beta)$ have an overlapping component $e \cap e^1$, and they bound the regions of two different arcs $\beta_0 \neq \beta$ of site s_β, they form an s_β-cycle C. But C is contained in $D_\mathcal{P}$, deriving a contradiction to Lemma 11. ◀

6 A randomized incremental algorithm

Consider a random permutation of the set of arcs $\mathcal{S}$, $o = (\alpha_1, \ldots, \alpha_h)$. For $1 \leq i \leq h$ define $\mathcal{S}_i = \{\alpha_1, \ldots, \alpha_i\} \subseteq \mathcal{S}$ to be the subset of the first i arcs in o. Given $\mathcal{S}_i$, let $\mathcal{P}_i$ denote a boundary curve for $\mathcal{S}_i$, which induces a domain $D_i = D_{\mathcal{P}_i}$.

The randomized algorithm is inspired by the randomized, two-phase, approach of Chew [7] for the Voronoi diagram of points in convex position; however, it constructs Voronoi-like diagrams of boundary curves $\mathcal{P}_i$ within a series of shrinking domains $D_i \supseteq D_{i+1}$. The boundary curves are obtained by the insertion operation, starting with $J(s, s_{\alpha_1})$, thus, they always admit a Voronoi-like diagram. In phase 1, the arcs in $\mathcal{S}$ get deleted one by one in reverse order of o, while recording the neighbors of each deleted arc at the time of its deletion. Let $\mathcal{P}_1 = \partial(D(s, s_{\alpha_1}) \cap D_\Gamma)$ and $D_1 = D(s, s_{\alpha_1}) \cap D_\Gamma$. Let $R(\alpha_1) = D_1$. $\mathcal{V}_l(\mathcal{P}_1) = \emptyset$ is the Voronoi-like diagram for $\mathcal{P}_1$. In phase 2, we start with $\mathcal{V}_l(\mathcal{P}_1)$ and incrementally compute $\mathcal{V}_l(\mathcal{P}_{i+1})$, $i = 1, \ldots, h-1$, by inserting arc α_{i+1} in $\mathcal{V}_l(\mathcal{P}_i)$, where $\mathcal{P}_{i+1} = \mathcal{P}_i \oplus \alpha_{i+1}$ and $\mathcal{V}_l(\mathcal{P}_{i+1}) = \mathcal{V}_l(\mathcal{P}_i) \oplus \alpha_{i+1}$. At the end we obtain $\mathcal{V}_l(\mathcal{P}_h)$, where $\mathcal{P}_h = \mathcal{S}$.

We have already established that $\mathcal{V}_l(\mathcal{S}) = \mathcal{V}(\mathcal{S})$ (Corollary 10) and $\mathcal{P}_h = \mathcal{S}$, thus, the algorithm is correct. Given the analysis and the properties of Voronoi-like diagrams established in Sections 3 and 4, as well as Lemma 21, the time analysis becomes similar to the one for the farthest-segment Voronoi diagram [10].

▶ **Lemma 29.** *$\mathcal{P}_i$ contains at most $2i$ arcs; thus, the complexity of $\mathcal{V}_l(\mathcal{P}_i)$ is $O(i)$.*

Proof. At each step of phase 2, one original arc is inserted and at most one additional arc is created by a split, thus, $|\mathcal{P}_i| \leq 2i$. The complexity of $\mathcal{V}_l(\mathcal{P}_i)$ is $O(|\mathcal{P}_i|)$, thus, it is $O(i)$. ◀

▶ **Lemma 30.** *The expected number of arcs in $\tilde{\mathcal{P}}_i$ (auxiliary boundary arcs and fine Γ-arcs) that are visited while inserting α_{i+1} is $O(1)$.*

Proof. To insert arc α_{i+1} at one step of phase 2, we may trace a number of arcs in $\tilde{\mathcal{P}}_i$ that may be auxiliary arcs and/or fine Γ-arcs between the pair of consecutive original arcs that has been stored with α_{i+1} in phase 1. Since every element of $\mathcal{S}_{i+1}$ is equally likely to be α_{i+1}, each pair of consecutive original arcs in $\mathcal{P}_{i+1}$ has probability $1/i$ to be considered at step i. Let n_j be the number of arcs inbetween the jth pair of original arcs in $\tilde{\mathcal{P}}_i$, $1 \leq j \leq i$; $\sum_{j=1}^{i} n_j = |\tilde{\mathcal{P}}_i|$ which is $O(i)$. The expected number of arcs that are traced is then $\sum_{j=1}^{i} n_j/i \in O(1)$. ◀

Using the same backwards analysis as in [10], we conclude with the following theorem.

▶ **Theorem 31.** *Given an abstract Voronoi diagram $\mathcal{V}(S)$, $\mathcal{V}(S \setminus \{s\}) \cap VR(s, S)$ can be computed in expected $O(h)$ time, where h is the complexity of $\partial VR(s, S)$. Thus, $\mathcal{V}(S \setminus \{s\})$ can also be computed in expected time $O(h)$.*

7 Concluding remarks

Updating an abstract Voronoi diagram, after deletion of one site, in deterministic linear time remains an open problem. We plan to investigate the applicability of Voronoi-like diagrams in the linear-time framework of Aggarwal et al. [1] in subsequent research.

The algorithms and the results in this paper (Theorem 31) are also applicable to concrete Voronoi diagrams of line segments and planar straight-line graphs (including simple and non-simple polygons) even though they do not strictly fall under the AVD model unless segments are disjoint. For intersecting line segments, $\partial\mathrm{VR}(s, S)$ is a Davenport-Schinzel sequence of order 4 [17] but this does not affect the complexity of the algorithm, which remains linear.

Examples of concrete diagrams that fall under the AVD umbrella and thus can benefit from our approach include [6]: disjoint line segments and disjoint convex polygons of constant size in the L_p norms, or under the Hausdorff metric; point sites in any convex distance metric or the Karlsruhe metric; additively weighted points that have non-enclosing circles; power diagrams with non-enclosing circles.

References

1. A. Aggarwal, L. Guibas, J. Saxe, and P. Shor. A linear-time algorithm for computing the Voronoi diagram of a convex polygon. *Discrete & Computational Geometry*, 4:591–604, 1989.
2. F. Aurenhammer, R. Klein, and D.-T. Lee. *Voronoi Diagrams and Delaunay Triangulations*. World Scientific, 2013. URL: `http://www.worldscientific.com/worldscibooks/10.1142/8685`.
3. C. Bohler, P. Cheilaris, R. Klein, C. H. Liu, E. Papadopoulou, and M. Zavershynskyi. On the complexity of higher order abstract Voronoi diagrams. *Computational Geometry: Theory and Applications*, 48(8):539–551, 2015. `doi:10.1016/j.comgeo.2015.04.008`.
4. C. Bohler, R. Klein, and C. Liu. Forest-like abstract Voronoi diagrams in linear time. In *Proc. 26th Canadian Conference on Computational Geometry (CCCG)*, 2014. URL: `http://www.cccg.ca/proceedings/2014/papers/paper20.pdf`.
5. C. Bohler, R. Klein, and C.-H. Liu. An Efficient Randomized Algorithm for Higher-Order Abstract Voronoi Diagrams. In *32nd International Symposium on Computational Geometry (SoCG)*, volume 51, pages 21:1–21:15, Dagstuhl, Germany, 2016. `doi:10.4230/LIPIcs.SoCG.2016.21`.
6. C. Bohler, C. H. Liu, E. Papadopoulou, and M. Zavershynskyi. A randomized divide and conquer algorithm for higher-order abstract Voronoi diagrams. *Computational Geometry: Theory and Applications*, 59(C):26–38, 2016. `doi:10.1016/j.comgeo.2016.08.004`.
7. L. P. Chew. Building Voronoi diagrams for convex polygons in linear expected time. Technical report, Dartmouth College, Hanover, USA, 1990.
8. F. Chin, J. Snoeyink, and C. A. Wang. Finding the medial axis of a simple polygon in linear time. *Discrete & Computational Geometry*, 21(3):405–420, 1999.
9. M. de Berg, O. Schwarzkopf, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2nd edition, 2000.
10. E. Khramtcova and E. Papadopoulou. An expected linear-time algorithm for the farthest-segment Voronoi diagram. arXiv:1411.2816v3 [cs.CG], 2017. Preliminary version in *Proc. 26th Int. Symp. on Algorithms and Computation (ISAAC), LNCS* 9472, 404-414, 2015.
11. R. Klein. *Concrete and Abstract Voronoi Diagrams*, volume 400 of *Lecture Notes in Computer Science*. Springer-Verlag, 1989.
12. R. Klein, E. Langetepe, and Z. Nilforoushan. Abstract Voronoi diagrams revisited. *Computational Geometry: Theory and Applications*, 42(9):885–902, 2009.
13. R. Klein and A. Lingas. Hamiltonian abstract Voronoi diagrams in linear time. In *Algorithms and Computation, 5th International Symposium, (ISAAC)*, volume 834 of *Lecture Notes in Computer Science*, pages 11–19, 1994.
14. R. Klein, K. Mehlhorn, and S. Meiser. Randomized incremental construction of abstract Voronoi diagrams. *Computational geometry: Theory and Applications*, 3:157–184, 1993.
15. K. Mehlhorn, S. Meiser, and R. Rasch. Furthest site abstract Voronoi diagrams. *International Journal of Computational Geometry and Applications*, 11(6):583–616, 2001.
16. A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. John Wiley, second edition, 2000.
17. E. Papadopoulou and M. Zavershynskyi. The higher-order Voronoi diagram of line segments. *Algorithmica*, 74(1):415–439, 2016. `doi:10.1007/s00453-014-9950-0`.
18. S. M. Preparata F.P. *Computational Geometry*. Texts and Monographs in Computer Science. Springer, New York, NY, 1985.
19. M. Sharir and P. K. Agarwal. *Davenport-Schinzel sequences and their geometric applications*. Cambridge university press, 1995.

From a $(p, 2)$-Theorem to a Tight (p, q)-Theorem

Chaya Keller[1]
Department of Mathematics, Ben Gurion University of the NEGEV
Be'er Sheva,Israel
kellerc@math.bgu.ac.il

Shakhar Smorodinsky
Department of Mathematics, Ben Gurion University of the NEGEV
Be'er Sheva, Israel
shakhar@math.bgu.ac.il

Abstract

A family $\mathcal{F}$ of sets is said to satisfy the (p, q)-property if among any p sets of $\mathcal{F}$ some q have a non-empty intersection. The celebrated (p, q)-theorem of Alon and Kleitman asserts that any family of compact convex sets in $\mathbb{R}^d$ that satisfies the (p, q)-property for some $q \geq d+1$, can be pierced by a fixed number (independent on the size of the family) $f_d(p, q)$ of points. The minimum such piercing number is denoted by $\mathsf{HD}_d(p, q)$. Already in 1957, Hadwiger and Debrunner showed that whenever $q > \frac{d-1}{d}p + 1$ the piercing number is $\mathsf{HD}_d(p, q) = p - q + 1$; no exact values of $\mathsf{HD}_d(p, q)$ were found ever since.

While for an arbitrary family of compact convex sets in $\mathbb{R}^d$, $d \geq 2$, a $(p, 2)$-property does not imply a bounded piercing number, such bounds were proved for numerous specific families. The best-studied among them is axis-parallel boxes in $\mathbb{R}^d$, and specifically, axis-parallel rectangles in the plane. Wegner (1965) and (independently) Dol'nikov (1972) used a $(p, 2)$-theorem for axis-parallel rectangles to show that $\mathsf{HD}_{\text{rect}}(p, q) = p - q + 1$ holds for all $q > \sqrt{2p}$. These are the only values of q for which $\mathsf{HD}_{\text{rect}}(p, q)$ is known exactly.

In this paper we present a general method which allows using a $(p, 2)$-theorem as a bootstrapping to obtain a tight (p, q)-theorem, for families with Helly number 2, even without assuming that the sets in the family are convex or compact. To demonstrate the strength of this method, we show that $\mathsf{HD}_{\text{d-box}}(p, q) = p - q + 1$ holds for all $q > c' \log^{d-1} p$, and in particular, $\mathsf{HD}_{\text{rect}}(p, q) = p - q + 1$ holds for all $q \geq 7 \log_2 p$ (compared to $q \geq \sqrt{2p}$, obtained by Wegner and Dol'nikov more than 40 years ago).

In addition, for several classes of families, we present improved $(p, 2)$-theorems, some of which can be used as a bootstrapping to obtain tight (p, q)-theorems. In particular, we show that any family $\mathcal{F}$ of compact convex sets in $\mathbb{R}^d$ with Helly number 2 admits a $(p, 2)$-theorem with piercing number $O(p^{2d-1})$, and thus, satisfies $\mathsf{HD}_{\mathcal{F}}(p, q) = p - q + 1$ for all $q > cp^{1-\frac{1}{2d-1}}$, for a universal constant c.

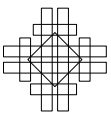
2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases (p, q)-Theorem, convexity, transversals, $(p, 2)$-theorem, axis-parallel rectangles

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.51

Related Version A full version of this paper is available at `https://arxiv.org/pdf/1712.04552.pdf`.

[1] The work of the first author was partially supported by the Shulamit Aloni Post-Doctoral Fellowship of the Israeli Ministry of Science and Technology and by the Kreitman Foundation Post-Doctoral Fellowship.



Funding This research was partially supported by Grant 635/16 from the Israel Science Foundation.

1 Introduction

1.1 Helly's theorem and (p,q)-theorems

The classical Helly's theorem says that if in a family of compact convex sets in $\mathbb{R}^d$ every $d+1$ members have a non-empty intersection then the whole family has a non-empty intersection.

For a pair of positive integers $p \geq q$, we say that a family $\mathcal{F}$ of sets satisfies the (p, q)-property if $|\mathcal{F}| \geq p$, none of the sets in $\mathcal{F}$ is empty, and among any p sets of $\mathcal{F}$ there are some q with a non-empty intersection. A set P is called a *transversal* (or alternatively, a *piercing set*) for $\mathcal{F}$ if it has a non-empty intersection with every member of $\mathcal{F}$. In this language, Helly's theorem states that any family of compact convex sets in $\mathbb{R}^d$ satisfying the $(d+1, d+1)$-property has a singleton transversal (alternatively, can be pierced by a single point).

In general, $d+1$ is clearly optimal in Helly's theorem, as any family of n hyperplanes in a general position in $\mathbb{R}^d$ satisfies the (d, d)-property but cannot be pierced by less than n/d points. However, for numerous specific classes of families, a (d', d')-property for some $d' < d+1$ is already sufficient to imply piercing by a single point. The minimal number d' for which this holds is called the *Helly number* of the family. For example, any family of *axis-parallel boxes* in $\mathbb{R}^d$ has Helly number 2.

In 1957, Hadwiger and Debrunner [13] proved the following generalization of Helly's theorem:

▶ **Theorem 1** (Hadwiger-Debrunner Theorem [13]). *For all $p \geq q \geq d+1$ such that $q > \frac{d-1}{d}p+1$, any family of compact convex sets in $\mathbb{R}^d$ that satisfies the (p, q)-property can be pierced by $p - q + 1$ points.*

▶ Remark. The bound in Theorem 1 is tight. Indeed, any family of n sets which consists of $p - q$ pairwise disjoint sets and $n - (p - q)$ copies of the same set satisfies the (p, q)-property but cannot be pierced by less than $p - q + 1$ points.

Hadwiger and Debrunner conjectured that while for general $p \geq q \geq d + 1$, a transversal of size $p - q + 1$ is not guaranteed, a (p, q)-property does imply a bounded-size transversal. This conjecture was proved only 35 years later, in the celebrated (p, q)-theorem of Alon and Kleitman.

▶ **Theorem 2** (Alon-Kleitman (p, q)-Theorem [2]). *For any triple of positive integers $p \geq q \geq d + 1$, there exists an integer $s = s(p, q, d)$ such that if $\mathcal{F}$ is a family of compact convex sets in $\mathbb{R}^d$ satisfying the (p, q)-property, then there exists a transversal for $\mathcal{F}$ of size at most s.*

The smallest value s that works for $p \geq q > d$ is called 'the Hadwiger-Debrunner number' and is denoted by $\mathsf{HD}_d(p, q)$. For various specific classes of families, a stronger (p, q)-theorem can be obtained. In such cases, we denote the minimal s that works for the family $\mathcal{F}$ by $\mathsf{HD}_{\mathcal{F}}(p, q)$.

The (p, q)-theorem has a rich history of variations and generalizations. To mention a few: In 1997, Alon and Kleitman [3] presented a simpler proof of the theorem (that leads to a somewhat weaker quantitative result). Alon et al. [1] proved in 2002 a 'topological' (p, q)-theorem for finite families of sets which are a *good cover* (i.e., the intersection of every

subfamily is either empty or contractible), and Bárány et al. [4] obtained in 2014 colorful and fractional versions of the theorem.

The size of the transversal guaranteed by the (p, q)-theorem is huge, and a large effort was invested in proving better bounds on $\mathsf{HD}_d(p, q)$, both in general and in specific cases. The most recent general result, by the authors and Tardos [16], shows that for any $\varepsilon > 0$, $\mathsf{HD}_d(p, q) \leq p - q + 2$ holds for all (p, q) such that $p > p_0(\varepsilon)$ and $q > p^{\frac{d-1}{d}+\varepsilon}$. Yet, no exact values of the Hadwiger-Debrunner number are known except for those given in the Hadwiger-Debrunner theorem. In fact, even the value $\mathsf{HD}_2(4, 3)$ is not known, the best bounds being $3 \leq \mathsf{HD}_2(4, 3) \leq 13$ (obtained by Kleitman et al. [18] in 2001).

1.2 $(p, 2)$-theorems and their applications

As mentioned above, while no general (p, q)-theorems exist for $q \leq d$, such theorems can be proved for various specific families. Especially desirable are $(p, 2)$-theorems, which relate the *packing number*, $\nu(\mathcal{F})$, of the family $\mathcal{F}$ (i.e., the maximum size of a subfamily all of whose members are pairwise disjoint) to its *piercing number*, $\tau(\mathcal{F})$ (i.e., the minimal size of a piercing set for the family $\mathcal{F}$).

In the last decades, $(p, 2)$-theorems were proved for numerous families. In particular, in 1991 Károlyi [15] proved a $(p, 2)$-theorem for axis-parallel boxes in $\mathbb{R}^d$, guaranteeing piercing by $O(p \log^{d-1} p)$ points. Kim et al. [17] proved in 2006 that any family of translates of a fixed convex set in $\mathbb{R}^d$ that satisfies the $(p, 2)$-property can be pierced by $2^{d-1}d^d(p - 1)$ points; five years later, Dumitrescu and Jiang [8] obtained a similar result for homothets of a convex set in $\mathbb{R}^d$. In 2012, Chan and Har-Peled proved a $(p, 2)$-theorem for families of pseudo-discs in the plane ([5], Theorem 4.6), with a piercing number linear in p. Two years ago, Govindarajan and Nivasch [11] showed that any family of convex sets in the plane in which among any p sets there is a pair that intersects on a given convex curve γ, can be pierced by $O(p^8)$ points.

In 2004, Matoušek [20] showed that families of sets with bounded dual VC-dimension have a bounded fractional Helly number. Recently, Pinchasi [21] has drawn a similar relation between the union complexity and the fractional Helly number. Each of these results implies a $(p, 2)$-theorem for the respective families, using the proof technique of the Alon-Kleitman (p, q)-theorem.

Besides their intrinsic interest, $(p, 2)$-theorems serve as a tool for obtaining other results. One such result is an *improved Ramsey Theorem*. Consider, for example, a family $\mathcal{F}$ of n axis-parallel rectangles in the plane. The classical Ramsey theorem implies that $\mathcal{F}$ contains a subfamily of size $\Omega(\log n)$, all whose elements are either pairwise disjoint or pairwise intersecting. As was observed by Larman et al. [19], the aforementioned $(p, 2)$-theorem for axis-parallel rectangles [15] allows obtaining an improved bound of $\Omega(\sqrt{n/\log n})$. Indeed, either $\mathcal{F}$ contains a subfamily of size $\left\lceil \sqrt{n/\log n} \right\rceil$ all whose elements are pairwise disjoint, and we are done, or $\mathcal{F}$ satisfies the $(p, 2)$-property with $p = \left\lceil \sqrt{n/\log n} \right\rceil$. In the latter case, by the $(p, 2)$-theorem, $\mathcal{F}$ can be pierced by $O(p \log p) = O(\sqrt{n \log n})$ points. The largest among the subsets of $\mathcal{F}$ pierced by a single point contains at least $\Omega(\frac{n}{\sqrt{n \log n}}) = \Omega(\sqrt{n/\log n})$ rectangles, and all its elements are pairwise intersecting.

Another result that can be obtained from a $(p, 2)$-theorem is an improved (p, q)-theorem; this will be described in detail below.

1.3 $(p, 2)$-theorems and (p, q)-theorems for axis-parallel rectangles and boxes

The (p, q)-problem for axis-parallel boxes is almost as old as the general (p, q)-problem, and was studied almost as thoroughly (see the survey of Eckhoff [9]). It was posed in 1960 by Hadwiger and Debrunner [14], who proved that any family of axis-parallel rectangles in the plane that satisfies the (p, q)-property, for $p \geq q \geq 2$, can be pierced by $\binom{p-q+2}{2}$ points. Unlike the (p, q)-problem for general families of convex sets, in this problem a finite bound on the piercing number was known from the very beginning, and the research goal has been to improve the bounds on this size, denoted $\mathsf{HD}_{\mathrm{rect}}(p, q)$ for rectangles and $\mathsf{HD}_{\mathrm{d-box}}(p, q)$ for boxes in $\mathbb{R}^d$.

For rectangles and $q = 2$, the quadratic upper bound on $\mathsf{HD}_{\mathrm{rect}}(p, 2)$ was improved to $O(p \log p)$ by Wegner (unpublished), and independently, by Károlyi [15]. The best currently known upper bound, which follows from a recursive formula presented by Fon Der Flaass and Kostochka [10], is

$$\mathsf{HD}_{\mathrm{rect}}(p, 2) \leq p\lceil \log_2 p \rceil - 2^{\lceil \log_2 p \rceil} + 1, \tag{1}$$

for all $p \geq 2$. On the other hand, it is known that the 'optimal possible' answer $p-q+1 = p-1$ fails already for $p = 4$. Indeed, Wegner [24] showed that $\mathsf{HD}_{\mathrm{rect}}(4, 2) = 5$, and by taking $\lceil p/3 \rceil - 1$ pairwise disjoint copies of his example, one obtains a family of axis-parallel rectangles that satisfies the $(p, 2)$-property but cannot be pierced by less than $\approx 5p/3$ points.

Wegner [24] conjectured that $\mathsf{HD}_{\mathrm{rect}}(p, 2)$ is linear in p, and is possibly even bounded by $2p - 3$. While Wegner's conjecture is believed to hold (see [9, 12]), no improvement of the bound (1) was found so far.

For rectangles and $q > 2$, Hadwiger and Debrunner showed that the exact bound $\mathsf{HD}_{\mathrm{rect}}(p, q) = p-q+1$ holds for all $q \geq p/2+1$. Wegner [24] and (independently) Dol'nikov [7] presented recursive formulas that allow leveraging a $(p, 2)$-theorem for axis-parallel rectangles into a tight (p, q)-theorem. Applying these formulas along with the Hadwiger-Debrunner quadratic upper bound on $\mathsf{HD}_{\mathrm{rect}}(p, 2)$, Dol'nikov showed that $\mathsf{HD}_{\mathrm{rect}}(p, q) = p - q + 1$ holds for all $2 \leq q \leq p < \binom{q+1}{2}$. Applying the formulas along with the improved bound (1) on $\mathsf{HD}_{\mathrm{rect}}(p, 2)$, Scheller ([22], see also [9]) obtained by a computer-aided computation upper bounds on the minimal p such that $\mathsf{HD}_{\mathrm{rect}}(p, q) = p - q + 1$ holds, for all $q \leq 12$. These values suggest that $\mathsf{HD}_{\mathrm{rect}}(p, q) = p - q + 1$ holds already for $q = \Omega(\log p)$. However, it appears that the method in which Dol'nikov proved a tight bound in the range $p < \binom{q+1}{2}$ does not extend to show a tight bound for all $q = \Omega(\log p)$ (even if (1) is employed), and in fact, no concrete improvement of Dol'nikov's result was presented (see the survey [9]).

Dol'nikov [7] claimed that if $\mathsf{HD}_{\mathrm{rect}}(p, 2)$ is linear in p as conjectured by Wegner [24], then one can deduce that $\mathsf{HD}_{\mathrm{rect}}(p, q) = p - q + 1$ holds for all $q \geq c$, for some constant c. Eckhoff [9] wrote that the proof of this claim presented in [7] is flawed, but it is plausible that the claim does hold. On the other direction, nothing is known about the minimal q for which $\mathsf{HD}_{\mathrm{rect}}(p, q) = p - q + 1$ may hold; in particular, it is not impossible that the optimal bound $\mathsf{HD}_{\mathrm{rect}}(p, q) = p - q + 1$ holds already for $q = 3$.

For axis-parallel boxes in $\mathbb{R}^d$, the aforementioned recursive formula of [10] implies the bound $\mathsf{HD}_{\mathrm{d-box}}(p, 2) \leq O(p \log^{d-1} p)$. While it is believed that the correct upper bound is $O(p)$, the result of [10] was not improved ever since; the only advancement is a recent result of Chudnovsky et al. [6], who proved an upper bound of $O(p \log \log p)$ for any family of axis-parallel boxes in which for each two intersecting boxes, a corner of one is contained in the other.

1.4 Our results

From $(p, 2)$-theorems to (p, q)-theorems

The main result of this paper is a general method for leveraging a $(p, 2)$-theorem into a tight (p, q)-theorem, applicable to families with Helly number 2. Interestingly, the method does not assume that the sets in $\mathcal{F}$ are convex or compact.

▶ **Theorem 3.** *For any $m \in \mathbb{N}$, there exists $c' = c'(m)$ such that the following holds. Let $\mathcal{F}$ be a family of sets in $\mathbb{R}^d$ such that $\mathsf{HD}_{\mathcal{F}}(2, 2) = 1$. Assume that for all $2 \leq p \in \mathbb{N}$ we have $\mathsf{HD}_{\mathcal{F}}(p, 2) \leq pf(p)$, where $f : [2, \infty) \to [1, \infty)$ is a differentiable function of p that satisfies $f'(p) \geq \frac{\log_2 e}{p}$ and $\frac{f'(p)}{f(p)} \leq \frac{m}{p}$ for all $p \geq 2$. Denote $T_c(p) = T_c(p, f) = \min\{q : q \geq 2c \cdot f(2p/q)\}$. Then for any $p \geq q \geq 2$ such that $q \geq T_{c'}(p)$, we have $\mathsf{HD}_{\mathcal{F}}(p, q) = p - q + 1$.*

While the condition on the function $f(p)$ looks a bit "scary", it actually holds for any function f whose growth rate (as expressed by its derivative $f'(p)$ and by the derivative of its logarithm $(\log f(p))' = \frac{f'(p)}{f(p)}$) is between the growth rates of $f(p) = \log_2 p$ and $f(p) = p^m$ (where m can be any integer, and c' in the assertion depends on it), including all cases needed in the current paper.

The first application of our general method is the following theorem for families of axis-parallel rectangles in the plane, obtained using (1) as the basic $(p, 2)$-theorem and some local refinements.

▶ **Theorem 4.** $\mathsf{HD}_{\mathrm{rect}}(p, q) = p - q + 1$ *holds for all* $q \geq 7 \log_2 p$.

▶ Remark. Theorem 4 improves significantly on the best previous result of Wegner (1965) and Dol'nikov (1972), that obtained the exact value $\mathsf{HD}_{\mathrm{rect}}(p, q) = p - q + 1$ only for $q > \sqrt{2p}$.

Another corollary is a tight (p, q)-theorem for axis-parallel boxes in $\mathbb{R}^d$:

▶ **Theorem 5.** $\mathsf{HD}_{\mathrm{d-box}}(p, q) = p - q + 1$ *holds for all* $q > c \log^{d-1} p$, *where c is a universal constant.*

In the proof of Theorem 3 we deploy the following observation of Wegner and Dol'nikov, which holds for any family $\mathcal{F}$ with Helly number 2:

$$\mathsf{HD}_{\mathcal{F}}(p, q) \leq \mathsf{HD}_{\mathcal{F}}(p - \lambda, q - 1) + \lambda - 1, \tag{2}$$

where $\lambda = \nu(\mathcal{F})$ is the packing number of $\mathcal{F}$ (Observation 8 below). We use an inductive process in which (2) is applied as long as $\mathcal{F}$ contains a sufficiently large pairwise-disjoint set. To treat the case where $\mathcal{F}$ does not contain a 'large' pairwise-disjoint set (and thus, $\nu(\mathcal{F})$ is small), we make use of a combinatorial argument, based on a variant of a 'combinatorial dichotomy' presented by the authors and Tardos [16], which first leverages the $(p, 2)$-theorem into a 'weak' (p, q)-theorem, and then uses that (p, q)-theorem to show that if $\nu(\mathcal{F})$ is 'small' then $\tau(\mathcal{F}) < p - q + 1$.

From $(2, 2)$-theorems to $(p, 2)$-theorems

It is natural to ask, under which conditions a $(2, 2)$-theorem implies a $(p, 2)$-theorem for all $p > 2$.

While in general, a $(2, 2)$-theorem does not imply a $(p, 2)$-theorem (see an example in the full version of the paper), we prove such an implication for several kinds of families. Our first result here concerns families with Helly number 2 (i.e., families $\mathcal{F}$ with $\mathsf{HD}_{\mathcal{F}}(2, 2) = 1$).

▶ **Theorem 6.** *Let $\mathcal{F}$ be a family of compact convex sets in $\mathbb{R}^d$ with Helly number 2. Then $\mathsf{HD}_{\mathcal{F}}(p, 2) \leq p^{2d-1}/2^{d-1}$, and consequently, $\mathsf{HD}_{\mathcal{F}}(p, q) = p - q + 1$ holds for all $q > cp^{1-\frac{1}{2d-1}}$, where $c = c(d)$ is a constant depending only on the dimension d.*

The second result only assumes the existence of a $(2, 2)$-theorem (where the piercing set may contain more than one point).

▶ **Theorem 7.** *Let $\mathcal{F}$ be a family of compact convex sets in $\mathbb{R}^d$ that admits a $(2, 2)$-theorem. Then:*

1. *$\mathcal{F}$ admits a $(p, 2)$-theorem for piercing with a bounded number $s = s(p, d)$ of points.*
2. *If $d = 2$, then $\mathsf{HD}_{\mathcal{F}}(p, 2) = O(p^8 \log^2 p)$.*
3. *If $d = 2$ and $\mathcal{F}$ has a bounded VC-dimension (see [23]), then $\mathsf{HD}_{\mathcal{F}}(p, 2) = O(p^4 \log^2 p)$.*

Since families with a sub-quadratic union complexity admit a $(2, 2)$-theorem and have a bounded VC-dimension, Theorem 7(3) implies that any family $\mathcal{F}$ of regions in the plane with a sub-quadratic union complexity satisfies $\mathsf{HD}_{\mathcal{F}}(p, 2) = O(p^4 \log^2 p)$. This significantly improves over the bound $\mathsf{HD}_{\mathcal{F}}(p, 2) = O(p^{16})$ that was obtained for such families in [16].

1.5 Organization of the paper

In Section 2 we demonstrate our general method for leveraging a $(p, 2)$-theorem into a tight (p, q)-theorem and prove Theorem 4. Our new $(p, 2)$-theorem for convex sets with Helly number 2 (i.e., Theorem 6 above) is presented in Section 3. We conclude the paper with a discussion and open problems in Section 4. For space reasons, the proofs of Theorems 3, 5, and 7 are presented only in the full version of the paper.

2 From $(p, 2)$-theorems to tight (p, q)-theorems

In this section we present our main theorem which allows leveraging a $(p, 2)$-theorem into a tight (p, q)-theorem, for families $\mathcal{F}$ that satisfy $\mathsf{HD}_{\mathcal{F}}(2, 2) = 1$. As the proof of the theorem in its full generality is somewhat complex, we present here the proof in the case of axis-parallel rectangles in the plane, and provide the full proof in the full version of the paper. Before presenting the proof of the theorem, we briefly present the Wegner-Dol'nikov argument (parts of which we use in our proof) in Section 2.1, provide an outline of our method in Section 2.2, and prove two preparatory lemmas in Section 2.3.

2.1 The Wegner-Dol'nikov method

As mentioned in the introduction, Wegner and (independently) Dol'nikov leveraged the Hadwiger-Debrunner $(p, 2)$-theorem for axis-parallel rectangles in the plane, which asserts that $\mathsf{HD}_{\mathrm{rect}}(p, 2) \leq \binom{p}{2}$, into a tight (p, q)-theorem, asserting that $\mathsf{HD}_{\mathrm{rect}}(p, q) \leq p - q + 1$ holds for all $p \geq q \geq 2$ such that $p < \binom{q+1}{2}$. The heart of the Wegner-Dol'nikov argument is the following observation.

▶ **Observation 8.** *Let $\mathcal{F}$ be a family that satisfies $\mathsf{HD}_{\mathcal{F}}(2, 2) = 1$, and put $\lambda = \nu(\mathcal{F})$. Then*

$$\mathsf{HD}_{\mathcal{F}}(p, q) \leq \mathsf{HD}_{\mathcal{F}}(p - \lambda, q - 1) + \lambda - 1.$$

Proof. The slightly weaker bound $\mathsf{HD}_{\mathcal{F}}(p, q) \leq \mathsf{HD}_{\mathcal{F}}(p - \lambda, q - 1) + \lambda$ holds trivially, and does not even require the assumption $\mathsf{HD}_{\mathcal{F}}(2, 2) = 1$. Indeed, if $\mathcal{S}$ is a pairwise-disjoint subset of $\mathcal{F}$ of size λ, then $\mathcal{F} \setminus \mathcal{S}$ satisfies the $(p - \lambda, q - 1)$-property, and thus, can be

pierced by $\mathsf{HD}_{\mathcal{F}}(p-\lambda, q-1)$ points. As $\mathcal{S}$ clearly can be pierced by λ points, we obtain $\mathsf{HD}_{\mathcal{F}}(p,q) \leq \mathsf{HD}_{\mathcal{F}}(p-\lambda, q-1) + \lambda$.

To get the improvement by 1, let $\mathcal{S}$ be a pairwise-disjoint subfamily of $\mathcal{F}$ of size $\lambda = \nu(\mathcal{F})$ and let T be a transversal of $\mathcal{F} \setminus \mathcal{S}$ of size $\mathsf{HD}_{\mathcal{F}}(p-\lambda, q-1)$. Take an arbitrary $x \in T$, and consider the subfamily $\mathcal{X} = \{A \in \mathcal{F} \setminus \mathcal{S} : x \in A\}$ (i.e., the sets in $\mathcal{F} \setminus \mathcal{S}$ pierced by x). By the maximality of $\mathcal{S}$, each $A \in \mathcal{X}$ intersects some $B \in \mathcal{S}$. Hence, we can write $\mathcal{X} = \cup_{B \in \mathcal{S}} \mathcal{X}_B$, where $\mathcal{X}_B = \{A \in \mathcal{X} : A \cap B \neq \emptyset\}$. Observe that for each B, the set $\mathcal{X}_B \cup \{B\}$ is pairwise-intersecting. Indeed, any $A, A' \in \mathcal{X}$ intersect in x, and all elements of $\mathcal{X}_B$ intersect B. Therefore, by the assumption on $\mathcal{F}$, each $\mathcal{X}_B \cup \{B\}$ can be pierced by a single point. Since $\mathcal{X} = \cup_{B \in \mathcal{S}} \mathcal{X}_B$, this implies that there exists a transversal T' of $\mathcal{X} \cup \mathcal{S}$ of size $|\mathcal{S}| = \lambda$. Now, the set $(T \setminus \{x\}) \cup T'$ is the desired transversal of $\mathcal{F}$ with $\mathsf{HD}_{\mathcal{F}}(p-\lambda, q-1) + \lambda - 1$ points. ◀

▶ Remark. We note that a generally similar argument of dividing the family $\mathcal{F}$ into a 'good' subfamily that satisfies a 'stronger' property (like the $(p-\lambda, q-1)$-property in Observation 8) and a small 'bad' subfamily that does not admit a 'good' property (like the independent set in Observation 8 which clearly cannot be pierced by less than λ points) appears also in the improved bound on the Hadwiger-Debrunner number for general convex sets presented in [16]. While in [16], the bound on the piercing number is $p - q + 2$, Observation 8 leads to the optimal piercing number $p - q + 1$, as shown below. The advantage of Observation 8 is the 'improvement by 1' step, which reduces the piercing number by 1; this step cannot be applied for general families of compact convex sets, since it relies on the fact that axis-parallel rectangles have Helly number 2.

Using Observation 8, Wegner and Dol'nikov proved the following theorem, which we will use in our proof below.

▶ **Theorem 9** ([7], Theorem 2; [24]). *Let $\mathcal{F}$ be a family of axis-parallel rectangles in the plane. Then for any $p \geq q \geq 2$ such that $p < \binom{q+1}{2}$, we have $\mathsf{HD}_{\mathcal{F}}(p,q) = p - q + 1$.*

Proof. The proof is by induction. The induction basis is $q = 2$: for this value, the assertion is relevant only for $p = 2$, and we indeed have $\mathsf{HD}_{\text{rect}}(2,2) = 1 = 2 - 2 + 1$ as asserted.

For the inductive step, we consider $\lambda = \nu(\mathcal{F})$. Note that $\mathcal{F}$ satisfies the $(\lambda+1, 2)$-property. Thus, if $\binom{\lambda+1}{2} \leq p - q + 1$ then we have $\mathsf{HD}_{\mathcal{F}}(p,q) \leq p - q + 1$ by the aforementioned Hadwiger-Debrunner $(p,2)$-theorem for axis-parallel rectangles. On the other hand, if $\binom{\lambda+1}{2} > p - q + 1$ then it can be checked that $p - \lambda < \binom{q}{2}$, so by the induction hypothesis we have $\mathsf{HD}_{\mathcal{F}}(p-\lambda, q-1) = (p-\lambda) - (q-1) + 1$. By Observation 8, this implies $\mathsf{HD}_{\mathcal{F}}(p,q) \leq \mathsf{HD}_{\mathcal{F}}(p-\lambda, q-1) + \lambda - 1 = p - q + 1$, as asserted. ◀

2.2 Outline of our method

Let $\mathcal{F}$ be a family of axis-parallel rectangles in the plane. Instead of leveraging the Hadwiger-Debrunner $(p,2)$-theorem for $\mathcal{F}$ into a (p,q)-theorem as was done by Wegner and Dol'nikov, we would like to leverage the stronger bound $\mathsf{HD}_{\text{rect}}(p,2) \leq p \log_2 p$ which follows from (1). We want to deduce that $\mathsf{HD}_{\text{rect}}(p,q) = p - q + 1$ holds for all $q \geq 7 \log p$.

Basically, we would like to perform an inductive process similar to the process applied in the proof of Theorem 9. As above, put $\lambda = \nu(\mathcal{F})$. If λ is 'sufficiently large' (namely, if $q - 1 \geq 7 \log_2(p-\lambda)$), we apply the recursive formula $\mathsf{HD}_{\mathcal{F}}(p,q) \leq \mathsf{HD}_{\mathcal{F}}(p-\lambda, q-1) + \lambda - 1$ and use the induction hypothesis to bound $\mathsf{HD}_{\mathcal{F}}(p-\lambda, q-1)$. Otherwise, we would like to use the improved $(p,2)$-theorem to deduce that $\mathcal{F}$ can be pierced by at most $p - q + 1$ points.

However, since we want to prove the theorem in the entire range $q \geq 7 \log_2 p$, in order to apply the induction hypothesis to $\mathsf{HD}_{\mathcal{F}}(p - \lambda, q - 1)$, λ must be at least *linear in* p (specifically, we need $\lambda \geq 0.1p$, as is shown below). Thus, in the 'otherwise' case we have to show that if $\lambda < 0.1p$, then $\mathcal{F}$ can be pierced by at most $p - q + 1$ points. If we merely use the fact that $\mathcal{F}$ satisfies the $(\lambda + 1, 2)$-property and apply the improved $(p, 2)$-theorem, we only obtain that $\mathcal{F}$ can be pierced by $O(p \log p)$ points – significantly weaker than the desired bound $p - q + 1$.

Instead, we use a more complex procedure, partially based on the following observation, presented in [16] (and called there a 'combinatorial dichotomy'):

▶ **Observation 10.** *Let $\mathcal{F}$ be a family that satisfies the (p, q)-property. For any $p' \leq p, q' \leq q$ such that $q' \leq p'$, either $\mathcal{F}$ satisfies the (p', q')-property, or there exists $S \subset \mathcal{F}$ of size p' that does not contain an intersecting q'-tuple. In the latter case, $\mathcal{F} \setminus S$ satisfies the $(p - p', q - q' + 1)$-property.*

First, we use Observation 10 to leverage the $(p, 2)$-theorem by an inductive process into a 'weak' (p, q)-theorem that guarantees piercing with $p - q + 1 + O(p)$ points, for all $q = \Omega(\log p)$. We then show that if $\lambda < 0.1p$ then $\mathcal{F}$ can be pierced by at most $p - q + 1$ points, by combining the weak (p, q)-theorem, another application of Observation 10, and a lemma which exploits the size of λ.

2.3 The two main lemmas used in the proof

Our first lemma leverages the $(p, 2)$-theorem $\mathsf{HD}_{\text{rect}}(p, 2) \leq p \log_2 p$ into a weak (p, q)-theorem, using Observation 10.

▶ **Lemma 11.** *Let $\mathcal{F}$ be a family of axis-parallel rectangles in the plane. Then for any $c > 0$ and for any $p \geq q \geq 2$ such that $q \geq c \log_2 p$, we have*

$$\mathsf{HD}_{\mathcal{F}}(p, q) \leq p - q + 1 + \frac{2p}{c}.$$

Proof. First, assume that both p and q are powers of 2. We perform an inductive process with $\ell = (\log_2 q) - 1$ steps, where we set $\mathcal{F}_0 = \mathcal{F}$ and $(p_0, q_0) = (p, q)$, and in each step i, we apply Observation 10 to a family $\mathcal{F}_{i-1}$ that satisfies the (p_{i-1}, q_{i-1})-property, with $(p', q') = (\frac{p_{i-1}}{2}, \frac{q_{i-1}}{2})$ which we denote by (p_i, q_i).

Consider Step i. By Observation 10, either $\mathcal{F}_{i-1}$ satisfies the $(p_i, q_i) = (\frac{p_{i-1}}{2}, \frac{q_{i-1}}{2})$-property, or there exists a 'bad' set S_i of size $\frac{p_{i-1}}{2}$ without an intersecting $\frac{q_{i-1}}{2}$-tuple, and the family $\mathcal{F}_{i-1} \setminus S_i$ satisfies the $(\frac{p_{i-1}}{2}, \frac{q_{i-1}}{2} + 1)$-property, and in particular, the $(\frac{p_{i-1}}{2}, \frac{q_{i-1}}{2})$-property. In either case, we are reduced to a family $\mathcal{F}_i$ (either $\mathcal{F}_{i-1}$ or $\mathcal{F}_{i-1} \setminus S_i$) that satisfies the (p_i, q_i)-property, to which we apply Step $i + 1$.

At the end of Step ℓ we obtain a family $\mathcal{F}_\ell$ that satisfies the $(2p/q, 2)$-property. (Note that the ratio between the left term and the right term remains constant along the way.) By the $(p, 2)$-theorem, $\mathcal{F}_\ell$ can be pierced by $\frac{2p}{q} \log_2 \left(\frac{2p}{q}\right)$ points. As $q \geq \max(c \log_2 p, 2)$, this implies that $\mathcal{F}_\ell$ can be pierced by

$$\frac{2p}{q} \log_2 \left(\frac{2p}{q}\right) \leq \frac{2p}{q} \log_2 p \leq \frac{2p}{c}$$

points.

In order to pierce $\mathcal{F}$, we also have to pierce the 'bad' sets S_i. In the worst case, in each step we have a bad set, and so we have to pierce $S = \cup_{i=1}^{\ell} S_i$. The size of S is

$|S| = \frac{p}{2} + \frac{p}{4} + \ldots + 2 + 1 = p - 1$. Since any family that satisfies the (p, q)-property also satisfies the $(p - k, q - k)$-property for any k, the family S contains an intersecting $(q - 1)$-tuple, which of course can be pierced by a single point. Hence, S can be pierced by $(p-1)-(q-1)+1 = p-q+1$ points. Therefore, in total $\mathcal{F}$ can be pierced by $p-q+1+2p/c$ points, as asserted.

Now, we have to deal with the case where p, q are not necessarily powers of 2, and thus, in some of the steps either p_{i-1} or q_{i-1} or both are not divisible by 2. It is clear from the proof presented above that if we can define (p_i, q_i) in such a way that in both cases (i.e., whether there is a 'bad' set or not), we have $\frac{p_i}{q_i} \leq \frac{p_{i-1}}{q_{i-1}}$, and also the total size of the bad sets (i.e., $|S|$) is at most p, the assertion can be deduced as above (as the ratio between the left term and the right term only decreases). We show that this can be achieved by a proper choice of (p_i, q_i) and a slight modification of the steps described above. Let

$$(p', q') = \left(\left\lfloor \frac{p_{i-1}}{2} \right\rfloor, \left\lceil \frac{q_{i-1}}{2} \right\rceil \right).$$

If $\mathcal{F}_{i-1}$ satisfies the (p', q')-property, we define $\mathcal{F}_i = \mathcal{F}_{i-1}$ and $(p_i, q_i) = (p', q')$. Otherwise, there exists a 'bad' set S_i of size p' that does not contain an intersecting q'-tuple, and the family $\mathcal{F}_{i-1} \setminus S_i$ satisfies the

$$(p_{i-1} - p', q_{i-1} - q' + 1) = \left(\left\lceil \frac{p_{i-1}}{2} \right\rceil, \left\lfloor \frac{q_{i-1}}{2} \right\rfloor + 1 \right)$$

property. In this case, we define $\mathcal{F}_i = \mathcal{F}_{i-1} \setminus S_i$ and $(p_i, q_i) = (p_{i-1} - p', q_{i-1} - q' + 1)$.

It is easy to check that in both cases we have $\frac{p_i}{q_i} \leq \frac{p_{i-1}}{q_{i-1}}$, and that $|S| \leq p - 1$ holds also with respect to the modified definition of the S_i's. Hence, the proof indeed can be completed, as above. ◀

Our second lemma is a simple upper bound on the piercing number of a family that satisfies the $(p, 2)$-property. We shall use it to show that if $\nu(\mathcal{F})$ is 'small', then we can save 'something' when piercing large subsets of $\mathcal{F}$.

▶ **Lemma 12.** *Any family $\mathcal{G}$ of m sets that satisfies the $(p, 2)$-property can be pierced by $\left\lfloor \frac{m+p-1}{2} \right\rfloor$ points.*

Proof. We perform the following simple recursive process. If $\mathcal{G}$ contains a pair of intersecting sets, pierce them by a single point and remove both of them from $\mathcal{G}$. Continue in this fashion until all remaining sets are pairwise disjoint. Then pierce each remaining set by a separate point.

As $\mathcal{G}$ satisfies the $(p, 2)$-property, the number of sets that remain in the last step is at most $p-1$ if $m-(p-1)$ is even and at most $p-2$ otherwise. In the former case, the resulting piercing set is of size at most $\frac{m-(p-1)}{2} + (p-1) = \frac{m+p-1}{2}$. In the latter case, the piercing set is of size at most $\frac{m-(p-2)}{2} + (p-2) = \frac{m+p-2}{2}$. Hence, in both cases the piercing set is of size at most $\left\lfloor \frac{m+p-1}{2} \right\rfloor$, as asserted. ◀

▶ Remark. The assertion of Lemma 12 is tight, as for a family $\mathcal{G}$ composed of $m-p+2$ lines in a general position in the plane and $p-2$ pairwise-disjoint segments that do not intersect any of the lines, we have $|\mathcal{G}| = m$, $\mathcal{G}$ satisfies the $(p, 2)$-property, and $\mathcal{G}$ clearly cannot be pierced by less than $\left\lfloor \frac{m+p-1}{2} \right\rfloor$ points.

▶ **Corollary 13.** *Let $\mathcal{F}$ be a family of sets in $\mathbb{R}^d$, and put $\lambda = \nu(\mathcal{F})$. Then any subset $S \subset \mathcal{F}$ can be pierced by at most $\left\lfloor \frac{|S|+\lambda}{2} \right\rfloor$ points.*

The corollary follows from the lemma immediately, as any such family $\mathcal{F}$ satisfies the $(\lambda + 1, 2)$-property.

2.4 Proof of Theorem 4

Now we are ready to present the proof of our main theorem, in the specific case of axis-parallel rectangles in the plane. Let us recall its statement.

▶ **Theorem 4.** *Let $\mathcal{F}$ be a family of axis-parallel rectangles in the plane. If $\mathcal{F}$ satisfies the (p, q)-property, for $p \geq q \geq 2$ such that $q \geq 7 \log_2 p$, then $\mathcal{F}$ can be pierced by $p - q + 1$ points.*

▶ Remark. We note that the parameters in the proof (e.g., the values of (p', q') in the inductive step) were chosen in a sub-optimal way, that is however sufficient to yield the assertion with the constant 7. (The straightforward choice $(p', q') = (0.5p, 0.5q)$ is not sufficient for that). The constant can be further optimized by a more careful choice of the parameters; however, it seems that in order to reduce it below 6, a significant change in the proof is needed.

Proof of Theorem 4. The proof is by induction.

Induction basis. One can assume that $q \geq 37$, as for any smaller value of q, there are no p's such that $7 \log_2 p \leq q \leq p$. For $q = 37$, the theorem is only relevant for $(p, q) = (37, 37)$, and in this case we clearly have $\mathsf{HD}_{\mathcal{F}}(p, q) = 1 = p - q + 1$. Generally speaking, this is a sufficient basis, since in the inductive step, the value of q is reduced by 1 every time. However, in the proof of the inductive step we would like to assume that p, q are 'sufficiently large'; hence, we use Theorem 9 as the induction basis in order to cover a larger range of small (p, q) values.

We observe that for $q \leq 70$, all relevant (p, q) pairs (i.e., all pairs for which $7 \log_2 p \leq q \leq p$) satisfy $p \leq \binom{q+1}{2}$. Hence, in this range we have $\mathsf{HD}_{\mathcal{F}}(p, q) = p - q + 1$ by Theorem 9. Therefore, we may assume that $q > 70$; we also may assume $q < \sqrt{2p}$ (as otherwise, the assertion follows from Theorem 9), and thus, $p > 2450$ and so (using again the assumption $q < \sqrt{2p}$), also $p > 35q$.

Inductive step. Put $\lambda = \nu(\mathcal{F})$. By Observation 8, we have $\mathsf{HD}_{\mathcal{F}}(p, q) \leq \mathsf{HD}_{\mathcal{F}}(p - \lambda, q - 1) + \lambda - 1$. We want λ to be sufficiently large, such that if (p, q) lies in the range covered by the theorem (i.e., if $q \geq 7 \log_2 p$), then $(p - \lambda, q - 1)$ also lies in the range covered by the theorem (i.e., $q - 1 \geq 7 \log_2(p - \lambda)$). Note that the condition $q \geq 7 \log_2 p$ is equivalent to $2^{q/7} \geq p$, which implies $2^{(q-1)/7} = \frac{2^{q/7}}{2^{1/7}} \geq 0.9p$. Hence, if $\lambda \geq 0.1p$ then $q - 1 \geq 7 \log_2(p - \lambda)$, and so we can deduce from the induction hypothesis that

$$\mathsf{HD}_{\mathcal{F}}(p, q) \leq \mathsf{HD}_{\mathcal{F}}(p - \lambda, q - 1) + \lambda - 1 \leq (p - \lambda) - (q - 1) + 1 + (\lambda - 1) = p - q + 1,$$

as asserted. Therefore, it is sufficient to prove that $\mathsf{HD}_{\mathcal{F}}(p, q) \leq p - q + 1$ holds when $\lambda < 0.1p$.

Under this assumption on λ, we apply Observation 10 to $\mathcal{F}$, with $(p', q') = (\lfloor 0.62p \rfloor, 0.5q)$. We have to consider two cases:

Case 1: $\mathcal{F}$ satisfies the (p', q')-property. By the assumption on (p, q), we have $q \geq 7 \log_2 p$, and thus, $0.5q \geq 3.5 \log_2 p \geq 3.5 \log_2 \lfloor 0.62p \rfloor$. Hence, by Lemma 11,

$$\mathsf{HD}_{\mathcal{F}}\left(\lfloor 0.62p \rfloor, 0.5q\right) \leq 0.62p - 0.5q + 1 + \frac{2}{3.5} \cdot 0.62p < 0.975p - 0.5q + 1 \leq p - q + 1,$$

where the last inequality holds because we may assume $q \leq 0.05p$, since $p > 35q$ as was written above. Thus, $\mathcal{F}$ can be pierced by at most $p - q + 1$ points, as asserted.

Case 2: $\mathcal{F}$ does not satisfy the (p', q')-property. In this case, there exists a 'bad' subfamily S of size $p' = \lfloor 0.62p \rfloor$ that does not contain an intersecting $0.5q$-tuple, and the family $\mathcal{F} \setminus S$ satisfies the $(\lceil 0.38p \rceil, 0.5q)$-property.

To pierce $\mathcal{F} \setminus S$, we use Lemma 11. Like above, we have $0.5q \geq 3.5 \log_2 \lceil 0.38p \rceil$, whence by Lemma 11,

$$\mathsf{HD}_{\mathcal{F}}(\lceil 0.38p \rceil, 0.5q) \leq 0.39p - 0.5q + 1 + \frac{2}{3.5} \cdot 0.39p < 0.613p - 0.5q + 1,$$

where the first inequality holds since we may assume $p \geq 100$ (as was written above), and thus, $\lceil 0.38p \rceil \leq 0.39p$.

To pierce the 'bad' subfamily S, we use Corollary 13, which implies that S can be pierced by

$$\left\lfloor \frac{1}{2}(|S| + \lambda) \right\rfloor \leq \frac{1}{2}(0.62p + 0.1p) = 0.36p$$

points. Therefore, in total $\mathcal{F}$ can be pierced by $(0.613p - 0.5q + 1) + 0.36p < 0.975p - 0.5q + 1$ points. Since we may assume $q \leq 0.05p$ (like above), this implies that $\mathcal{F}$ can be pierced by $p - q + 1$ points. This completes the proof. ◀

3 From $(2, 2)$-theorems to $(p, 2)$-theorems

As was mentioned in the introduction, in general, the existence of a $(2, 2)$-theorem (and even Helly number 2) does not imply the existence of a $(p, 2)$-theorem. An example mentioned by Fon der Flaass and Kostochka [10] (in a slightly different context) is presented in the full version of the paper.

In this section we prove Theorem 6 which asserts that for families of convex sets with Helly number 2, a $(2, 2)$-theorem does imply a $(p, 2)$-theorem, and consequently, a tight (p, q)-theorem for a large range of q's. Due to space constraints, the proof of our other new $(p, 2)$-theorem (i.e., Theorem 7) is presented in the full version of the paper.

Let us recall the assertion of the theorem:

▶ **Theorem 6.** *For any family $\mathcal{F}$ of compact convex sets in $\mathbb{R}^d$ that has Helly number 2, we have $\mathsf{HD}_{\mathcal{F}}(p, 2) \leq \frac{p^{2d-1}}{2^{d-1}}$. Consequently, we have $\mathsf{HD}_{\mathcal{F}}(p, q) = p - q + 1$ for all $q > cp^{1-\frac{1}{2d-1}}$, where $c = c(d)$.*

The 'consequently' part follows immediately from the $(p, 2)$-theorem via Theorem 3. Hence, we only have to prove the $(p, 2)$-theorem.

Let us present the proof idea first. The proof goes by induction on d. Given a family $\mathcal{F}$ of sets in $\mathbb{R}^d$ that satisfies the assumptions of the theorem and has the $(p, 2)$-property, we take $\mathcal{S}$ to be a maximum (with respect to size) pairwise-disjoint subfamily of $\mathcal{F}$, and consider the intersections of other sets of $\mathcal{F}$ with the elements of $\mathcal{S}$. We observe that by the maximality of $\mathcal{S}$, each set $A \in \mathcal{F} \setminus \mathcal{S}$ intersects at least one element of $\mathcal{S}$, and thus, we may partition $\mathcal{F}$ into three subfamilies: $\mathcal{S}$ itself, the family $\mathcal{U}$ of sets in $\mathcal{F} \setminus \mathcal{S}$ that intersect only one element of $\mathcal{S}$, and the family $\mathcal{M} \subset \mathcal{F} \setminus \mathcal{S}$ of sets that intersect at least two elements of $\mathcal{S}$.

We show (using the maximality of $\mathcal{S}$ and the $(2, 2)$-theorem on $\mathcal{F}$) that $\mathcal{U} \cup \mathcal{S}$ can be pierced by $p - 1$ points. As for $\mathcal{M}$, we represent it as a union of families: $\mathcal{M} = \cup_{C,C' \in \mathcal{S}} \mathcal{X}_{C,C'}$, where each $\mathcal{X}_{C,C'}$ consists of the elements of $F \setminus \mathcal{S}$ that intersect both C and C'. We use a geometric argument to show that each $\mathcal{X}_{C,C'}$ corresponds to $\mathcal{Y}_{C,C'} \subset \mathbb{R}^{d-1}$ that has Helly number 2 and satisfies the $(p, 2)$-property. This allows us to bound the piercing number of

$\mathcal{Y}_{C,C'}$ by the induction hypothesis, and consequently, to bound the piercing number of $\mathcal{X}_{C,C'}$. Adding up the piercing numbers of all $\mathcal{X}_{C,C'}$'s and the piercing number of $\mathcal{U} \cup \mathcal{S}$ completes the inductive step.

Proof of Theorem 6. By induction on d.

Induction basis. For any family $\mathcal{F}$ of compact convex sets in $\mathbb{R}^1$, by the Hadwiger-Debrunner theorem [13] we have $\mathsf{HD}_{\mathcal{F}}(p, 2) = p - 2 + 1 < p = p^{2 \cdot 1 - 1}/2^{1-1}$, and so the assertion holds.

Inductive step. Let $\mathcal{F}$ be a family of sets in $\mathbb{R}^d$ that satisfies the assumptions of the theorem and has the $(p, 2)$-property. Let $\mathcal{S}$ be a maximum (with respect to size) pairwise-disjoint subfamily of $\mathcal{F}$. We may assume $|\mathcal{S}| = p - 1$.

By the maximality of $\mathcal{S}$, each set $A \in \mathcal{F} \setminus \mathcal{S}$ intersects at least one element of $\mathcal{S}$. Moreover, any two sets $A, B \in \mathcal{F}$ that intersect the same $C \in \mathcal{S}$ and do not intersect any other element of $\mathcal{S}$, are intersecting, as otherwise, the subfamily $\mathcal{S} \cup \{A, B\} \setminus \{C\}$ would be a pairwise-disjoint subfamily of $\mathcal{F}$ that is larger than $\mathcal{S}$, a contradiction. Hence, for each $C_0 \in \mathcal{S}$, the subfamily

$$\mathcal{X}_{C_0} = \{A \in \mathcal{F} : \{C \in \mathcal{S} : A \cap C \neq \emptyset\} = \{C_0\}\} \cup \{C_0\}$$

satisfies the $(2, 2)$-property, and thus, can be pierced by a single point by the assumption on $\mathcal{F}$. Therefore, denoting $\mathcal{U} = \{A \in \mathcal{F} : |\{C \in \mathcal{S} : A \cap C \neq \emptyset\}| = 1\}$, all sets in $\mathcal{U} \cup \mathcal{S}$ can be pierced by at most $p - 1$ points.

Let $\mathcal{M} \subset \mathcal{F}$ be the family of all sets in $\mathcal{F}$ that intersect at least two elements of $\mathcal{S}$. For each $C, C' \in \mathcal{S}$, let

$$\mathcal{X}_{C,C'} = \{A \in \mathcal{F} \setminus \mathcal{S} : A \cap C \neq \emptyset \wedge A \cap C' \neq \emptyset\}.$$

(Note that the elements of $\mathcal{X}_{C,C'}$ may intersect other elements of $\mathcal{S}$.) Let $H \subset \mathbb{R}^d$ be a hyperplane that strictly separates C from C', and put $\mathcal{Y}_{C,C'} = \{A \cap H : A \in \mathcal{X}_{C,C'}\}$.

▶ **Claim 14.** *$\mathcal{Y}_{C,C'} \subset H \approx \mathbb{R}^{d-1}$ admits $\mathsf{HD}_{\mathcal{Y}_{C,C'}}(2, 2) = 1$ and satisfies the $(p, 2)$-property.*

Proof. To prove the claim, we observe that $A \cap H, A' \cap H \in \mathcal{Y}_{C,C'}$ intersect if and only if A and A' intersect. Indeed, assume $A \cap A' \neq \emptyset$. The family $\{A, A', C\}$ satisfies the $(2, 2)$-property, and hence, can be pierced by a single point by the assumption on $\mathcal{F}$. Thus, $A \cap A'$ contains a point of C. For the same reason, $A \cap A'$ contains a point of C'. Therefore, $A \cap A'$ contains points on the two sides of the hyperplane H. However, $A \cap A'$ is convex, and so, $(A \cap A') \cap H \neq \emptyset$, which means that $(A \cap H)$ and $(A' \cap H)$ intersect. The other direction is obvious.

It is now clear that as $\mathcal{X}_{C,C'} \subset \mathcal{F}$ satisfies the $(p, 2)$-property, $\mathcal{Y}_{C,C'}$ satisfies the $(p, 2)$-property as well. Moreover, let $T = \{A_1 \cap H, A_2 \cap H, A_3 \cap H, \ldots\} \subset \mathcal{Y}_{C,C'}$ be pairwise-intersecting. The corresponding family $\tilde{T} = \{C, A_1, A_2, A_3, \ldots\}$ is pairwise-intersecting, and thus, can be pierced by a single point by the assumption on $\mathcal{F}$. Thus, $(A_1 \cap A_2 \cap A_3 \cap \ldots) \cap C \neq \emptyset$. For the same reason, $(A_1 \cap A_2 \cap A_3 \cap \ldots) \cap C' \neq \emptyset$. Since $A_1 \cap A_2 \cap A_3 \cap \ldots$ is convex, this implies that $(A_1 \cap A_2 \cap A_3 \cap \ldots) \cap H \neq \emptyset$, or equivalently, that the family T can be pierced by a single point. Therefore, $\mathcal{Y}_{C,C'}$ satisfies $\mathsf{HD}_{\mathcal{Y}_{C,C'}}(2, 2) = 1$, as asserted. ◀

Claim 14 allows us to apply the induction hypothesis to $\mathcal{Y}_{C,C'}$, to deduce that it can be pierced by less than $p^{2d-3}/2^{d-1}$ points. Since $\mathcal{S}$ contains only $\binom{p-1}{2}$ pairs (C, C'), and since any set in $\mathcal{M}$ belongs to at least one of the $\mathcal{X}_{C,C'}$, this implies that $\mathcal{M}$ can be pierced by

less than $\binom{p-1}{2} \cdot p^{2d-3}/2^{d-2}$ points. As $\mathcal{U} \cup \mathcal{S}$ can be pierced by $p-1$ points as shown above, $\mathcal{F}$ can be pierced by less than

$$\binom{p-1}{2} \cdot \frac{p^{2d-3}}{2^{d-2}} + (p-1) < \frac{p^{2d-1}}{2^{d-1}}$$

points. This completes the proof. ◀

4 Discussion and open problems

A central problem left for further research is whether Theorem 3 which allows leveraging a $(p,2)$-theorem into a (p,q)-theorem, can be extended to the cases $\mathsf{HD}_{\mathcal{F}}(p,2) = pf(p)$ where $f(p) \ll \log p$ or $f(p)$ being super-polynomial in p. It seems that super-polynomial growth rates can be handled with a slight modification of the argument (at the expense of replacing $T_{c'}(p)$ with some worse dependence on p). For a sub-logarithmic growth rate, it seems that the current argument does not work, since the inductive step requires the packing number of $\mathcal{F}$ to be extremely large, and so, Lemma 12 allows reducing the piercing number of the 'bad' family S only slightly, rendering Lemma 11 insufficient for piercing $\mathcal{F}$ with $p-q+1$ points in total.

Extending the method for sub-logarithmic growth rates will have interesting applications. For instance, it will immediately yield a tight (p,q)-theorem for all $q = \Omega(\log\log p)$ for families of axis-parallel boxes in which for each two intersecting boxes, a corner of one is contained in the other, following the work of Chudnovsky et al. [6]. Furthermore, it will imply that if axis-parallel rectangles admit a $(p,2)$-theorem with the size of the piercing set linear in p (as conjectured by Wegner [24]), then $\mathsf{HD}_{\text{rect}}(p,q) = p-q+1$ holds for all $q \geq c$ for a constant c. As mentioned in Section 1.3, this was claimed by Dol'nikov [7], but with a flawed argument, as remarked by Eckhoff [9].

Another open problem is whether the method can be extended to families $\mathcal{F}$ that admit a $(2,2)$-theorem, but satisfy $\mathsf{HD}_{\mathcal{F}}(2,2) > 1$. Such an improvement would allow transformation into tight (p,q)-theorems of the $(p,2)$-theorems presented in Section 1.3, such as the $(p,2)$-theorem for pseudo-discs of Chan and Har-Peled [5].

A main obstacle here is that in this case, Observation 8 does not apply, and instead, we have the bound $\mathsf{HD}(p,q) \leq \mathsf{HD}(p-\lambda, q-1) + \lambda$. While the bound is only slightly weaker, it precludes us from using the inductive process of Wegner and Dol'nikov, as in each application of the inductive step we have an 'extra' point.

References

1 N. Alon, G. Kalai, R. Meshulam, and J. Matoušek. Transversal numbers for hypergraphs arising in geometry. *Adv. Appl. Math*, 29:79–101, 2002.

2 N. Alon and D.J. Kleitman. Piercing convex sets and the Hadwiger-Debrunner (p,q)-problem. *Advances in Mathematics*, 96(1):103–112, 1992. doi:10.1016/0001-8708(92)90052-M.

3 N. Alon and D.J. Kleitman. A purely combinatorial proof of the Hadwiger-Debrunner (p,q) conjecture. *Electr. J. Comb.*, 4(2), 1997.

4 I. Bárány, F. Fodor, L. Montejano, D. Oliveros, and A. Pór. Colourful and fractional (p, q)-theorems. *Discrete & Computational Geometry*, 51(3):628–642, 2014. doi:10.1007/s00454-013-9559-0.

5 T.M. Chan and S. Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. *Discrete & Computational Geometry*, 48(2):373–392, 2012. doi:10.1007/s00454-012-9417-5.

6 M. Chudnovsky, S. Spirkl, and S. Zerbib. Piercing axis-parallel boxes, Electron. J. Comb., to appear, 2017.

7 V. L. Dol'nikov. A certain coloring problem. *Sibirsk. Mat. Ž.*, 13:1272–1283, 1420, 1972.

8 A. Dumitrescu and M. Jiang. Piercing translates and homothets of a convex body. *Algorithmica*, 61(1):94–115, 2011. doi:10.1007/s00453-010-9410-4.

9 J. Eckhoff. A survey of the Hadwiger-Debrunner (p,q)-problem. In B. Aronov, S. Basu, J. Pach, and M. Sharir, editors, *Discrete and Computational Geometry*, volume 25 of *Algorithms and Combinatorics*, pages 347–377. Springer Berlin Heidelberg, 2003. doi:10.1007/978-3-642-55566-4_16.

10 D. Fon-Der-Flaass and A. V. Kostochka. Covering boxes by points. *Disc. Math.*, 120(1-3):269–275, 1993. doi:10.1016/0012-365X(93)90587-J.

11 S. Govindarajan and G. Nivasch. A variant of the Hadwiger-Debrunner (p, q)-problem in the plane. *Discrete & Computational Geometry*, 54(3):637–646, 2015. doi:10.1007/s00454-015-9723-9.

12 A. Gyárfás and J. Lehel. Covering and coloring problems for relatives of intervals. *Discrete Mathematics*, 55(2):167–180, 1985. doi:10.1016/0012-365X(85)90045-7.

13 H. Hadwiger and H. Debrunner. Über eine variante zum Hellyschen satz. *Archiv der Mathematik*, 8(4):309–313, 1957. doi:10.1007/BF01898794.

14 H. Hadwiger and H. Debrunner. *Combinatorial geometry in the plane.* Translated by V. Klee. With a new chapter and other additional material supplied by the translator. Holt, Rinehart and Winston, New York, 1964.

15 Gy. Károlyi. On point covers of parallel rectangles. *Period. Math. Hungar.*, 23(2):105–107, 1991. URL: https://doi-org.proxy1.athensams.net/10.1007/BF02280661.

16 C. Keller, S. Smorodinsky, and G. Tardos. On max-clique for intersection graphs of sets and the hadwiger-debrunner numbers. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017*, pages 2254–2263, 2017. doi:10.1137/1.9781611974782.148.

17 S. J. Kim, K. Nakprasit, M. J. Pelsmajer, and J. Skokan. Transversal numbers of translates of a convex body. *Discrete Mathematics*, 306(18):2166–2173, 2006. doi:10.1016/j.disc.2006.05.014.

18 D.J. Kleitman, A. Gyárfás, and G. Tóth. Convex sets in the plane with three of every four meeting. *Combinatorica*, 21(2):221–232, 2001. doi:10.1007/s004930100020.

19 D. Larman, J. Matoušek, J. Pach, and J. Töröcsik. A Ramsey-type result for planar convex sets. *Bulletin of London Math. Soc.*, 26:132–136, 1994.

20 J. Matoušek. Bounded VC-dimension implies a fractional Helly theorem. *Discrete & Computational Geometry*, 31(2):251–255, 2004. doi:10.1007/s00454-003-2859-z.

21 R. Pinchasi. A note on smaller fractional Helly numbers. *Discrete and Computational Geometry*, 54(3):663–668, 2015.

22 N. Scheller. (p,q)-probleme für quaderfamilien. Master's thesis, Universität Dortmund, 1996.

23 V. N. Vapnik and A. Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.

24 G. Wegner. Über eine kombinatorisch-geometrische Frage von Hadwiger und Debrunner. *Israel J. Math.*, 3:187–198, 1965. URL: https://doi-org.proxy1.athensams.net/10.1007/BF03008396.

Coloring Intersection Hypergraphs of Pseudo-Disks

Balázs Keszegh
Alfréd Rényi Institute of Mathematics, Hungarian Academy of Sciences and
MTA-ELTE Lendület Combinatorial Geometry Research Group
Budapest, Hungary
keszegh.balazs@renyi.mta.hu
https://orcid.org/0000-0002-3839-5103

Abstract

We prove that the intersection hypergraph of a family of n pseudo-disks with respect to another family of pseudo-disks admits a proper coloring with 4 colors and a conflict-free coloring with $O(\log n)$ colors. Along the way we prove that the respective Delaunay-graph is planar. We also prove that the intersection hypergraph of a family of n regions with linear union complexity with respect to a family of pseudo-disks admits a proper coloring with constantly many colors and a conflict-free coloring with $O(\log n)$ colors. Our results serve as a common generalization and strengthening of many earlier results, including ones about proper and conflict-free coloring points with respect to pseudo-disks, coloring regions of linear union complexity with respect to points and coloring disks with respect to disks.

2012 ACM Subject Classification Mathematics of computing → Hypergraphs

Keywords and phrases combinatorial geometry, conflict-free coloring, geometric hypergraph coloring

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.52

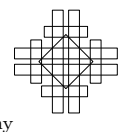
Related Version A full version of this paper is available at `https://arxiv.org/abs/1711.05473`.

Funding Research supported by the Lendület program of the Hungarian Academy of Sciences (MTA), under the grant LP2017-19/2017 and by the National Research, Development and Innovation Office – NKFIH under the grant K 116769.

Acknowledgements This paper is dedicated to my grandfather who died the same day when I found the proof for Theorem 14 (at that time with a non-explicit constant). Thanks to Dömötör Pálvölgyi for the inspiring discussions about these and other problems about coloring geometric hypergraphs defined by pseudo-disks. Thanks to Chaya Keller and the anonymous reviewers for many useful remarks about the realization of this paper.

1 Introduction

Proper colorings of hypergraphs and graphs defined by geometric regions are studied extensively due to their connections to, among others, conflict-free colorings and to cover-decomposability problems, both of which have real-life motivations in cellular networks, scheduling, etc. For applications and history we refer to the surveys [20, 26]. Here we restrict our attention to the (already long list of) results that directly precede our results. We discuss further directions and connections to other problems in Section 4.

34th International Symposium on Computational Geometry (SoCG 2018).
Editors: Bettina Speckmann and Csaba D. Tóth; Article No. 52; pp. 52:1–52:15


Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Our central family of regions is the family of pseudo-disks. Families of pseudo-disks have been regarded in many settings for a long time due to being the natural way of generalizing disks while retaining many of their topological and combinatorial properties. Problems regarded range from classic algorithmic questions like finding maximum size independent (disjoint) subfamilies [5] to classical combinatorial geometric questions like the Erdős-Szekeres problem [9]. Probably the most important pseudo-disk family is the family of homothets of a convex region. Pseudo-disks are also central in the area of geometric hypergraph coloring problems as we will see soon in this section.

Before we state our results and their precursors, we need to introduce some basic definitions.

▶ **Definition 1.** Given a hypergraph $\mathcal{H}$, a **proper coloring** of its vertices is a coloring in which every hyperedge H of size at least 2 of $\mathcal{H}$ contains two differently colored vertices.

Throughout the paper hypergraphs can contain hyperedges of size at least 2 only, in other words we do not allow (or automatically delete, when necessary) hyperedges of size 1.

▶ **Definition 2.** Given a hypergraph $\mathcal{H}$, a **conflict-free coloring** of its vertices is a coloring in which every hyperedge H contains a vertex whose color differs from the color of every other vertex of H.

▶ **Definition 3.** Given a family $\mathcal{B}$ of regions and a family $\mathcal{F}$ of regions, the **intersection hypergraph** of $\mathcal{B}$ **with respect to** (wrt. in short) $\mathcal{F}$ is the simple (that is, it has no multiple edges) hypergraph $\mathcal{I}(\mathcal{B},\mathcal{F})$ which has a vertex v_B corresponding to every $B \in \mathcal{B}$ and for every $F \in \mathcal{F}$ it has a hyperedge $H = \{v_B : B \cap F \neq \emptyset\}$ (if this set has size at least two) which corresponds to F. The **Delaunay-graph** of $\mathcal{B}$ wrt. $\mathcal{F}$ is the graph on the same vertex set containing only the hyperedges of $\mathcal{I}(\mathcal{B},\mathcal{F})$ of size 2.

Note that if $\mathcal{B}$ is finite, then even if $\mathcal{F}$ is infinite, $\mathcal{I}(\mathcal{B},\mathcal{F})$ has finitely many hyperedges. In particular, a hyperedge H can correspond to multiple members of $\mathcal{F}$.

We also define the following subgraph of the Delaunay-graph:

▶ **Definition 4.** The **restricted Delaunay-graph** of $\mathcal{B}$ wrt. $\mathcal{F}$ is the subgraph of the Delaunay-graph containing only those (hyper)edges $H_F = \{v_{B_1}, v_{B_2}\}$ for which the corresponding $F \in \mathcal{F}$ intersects B_1 and B_2 in disjoint regions, that is, $F \cap B_1 \neq \emptyset$, $F \cap B_2 \neq \emptyset$, $F \cap B_1 \cap B_2 = \emptyset$ and $F \cap B = \emptyset$ for every $B \in \mathcal{B} \setminus \{B_1, B_2\}$.

▶ **Observation 5** ([25]). *If $\mathcal{B}$ and $\mathcal{F}$ are families for which every hyperedge of $\mathcal{I}(\mathcal{B},\mathcal{F})$ contains a hyperedge of size 2, then a proper coloring of the Delaunay-graph is also a proper coloring of $\mathcal{I}(\mathcal{B},\mathcal{F})$.*

In the literature hypergraphs that have the property assumed in Observation 5 are sometimes called *rank two* hypergraphs (e.g., in [25]).

▶ **Definition 6.** We say that $\mathcal{B}$ can be properly/conflict-free/etc. colored wrt. $\mathcal{F}$ with $f(n)$ colors if for any $\mathcal{B}'$ subfamily of $\mathcal{B}$ of size n, the hypergraph $\mathcal{I}(\mathcal{B}',\mathcal{F})$ can be properly/conflict-free/etc. colored with $f(n)$ colors.

We note that one could further generalize this notion such that instead of intersection we regard inclusion/reverse-inclusion as the relation that defines the hyperedges, as it is done, e.g., in [16] for hypergraphs defined by intervals. As we do not consider problems in this paper for which this generalization is necessary, for simplicity we do not complicate our notation to contain these cases. For related problems see Section 4.

In the majority of earlier results, coloring points wrt. a family of regions or coloring a family of regions wrt. points were regarded. While they were not defined as intersection hypergraphs, they do fit into our general definition of intersection hypergraphs, choosing either $\mathcal{B}$ or $\mathcal{F}$ to be the family of points of the plane. Notice that whenever every point is or can be added as a member of the family $\mathcal{F}$, e.g., for disks, for homothets of a convex region and for pseudo-disks[1], coloring intersection hypergraphs of $\mathcal{F}$ wrt. to $\mathcal{F}$ is a common generalization both of coloring points wrt. $\mathcal{F}$ and coloring $\mathcal{F}$ wrt. points.

There are a few earlier results regarding this general setting. In [16] intersection (and also inclusion and reverse-inclusion) hypergraphs of intervals of the line were considered. In [11] and [8] they considered intersection hypergraphs (and graphs) of (unit) disks, pseudo-disks, squares and axis-parallel rectangles.

1.1 Results related to pseudo-disks

It is well known that the Delaunay-graph of points wrt. disks is planar and thus proper 4-colorable, and Observation 5 implies that the respective hypergraph is also proper 4-colorable.

In [25] Smorodinsky developed a general framework (based on the framework presented in [7]): using proper colorings of the subhypergraphs of the hypergraph with constantly many colors it can build a conflict-free coloring with $O(\log n)$ colors (where n is the number of vertices of the hypergraph). The results mentioned from now on use this framework to get a conflict-free coloring once there is a proper coloring. First, using the precursor of this framework, it was proved by Even et al. [7] that points wrt. disks admit a conflict-free coloring with $O(\log n)$ colors:

▶ **Theorem 7** ([7]). *Let $\mathcal{D}$ be the family of disks in the plane and S a finite set of points, $\mathcal{I}(S, \mathcal{D})$ admits a conflict-free coloring with $O(\log n)$ colors, where $n = |S|$.*

▶ **Definition 8.** A **Jordan region** is a (simply connected) closed bounded region whose boundary is a closed simple Jordan curve.

A family of Jordan regions is called a family of **pseudo-disks** if the boundaries of every pair of the regions intersect in at most two points.

Although we could not find it in the literature, it is well known that for pseudo-disks the bound of Theorem 7 holds as well, the reason is that the Delaunay-graph of points with respect to pseudo-disks is also a planar graph (implied by, e.g., Lemma 29, see later) and thus proper 4-colorable and then we can apply Observation 5 (we can assume that the hypergraph is rank two by, e.g., Corollary 26) and the same general framework to conclude the $O(\log n)$ upper bound for a conflict-free coloring.

The dual of Theorem 7 was also proved by Even et al. and was generalized to pseudo-disks by Smorodinsky:

▶ **Theorem 9** ([7]). *Let P be the set of all points of the plane and $\mathcal{B}$ a finite family of disks, $\mathcal{I}(\mathcal{B}, P)$ admits a proper coloring with 4 colors and a conflict-free coloring with $O(\log n)$ colors, where $n = |\mathcal{B}|$.*

▶ **Theorem 10** ([25]). *Let P be the set of all points of the plane and $\mathcal{B}$ be a finite family of pseudo-disks, $\mathcal{I}(\mathcal{B}, P)$ admits a proper coloring with a constant number of colors and a conflict-free coloring with $O(\log n)$ colors, where $n = |\mathcal{B}|$.*

[1] See Corollary 26 for a precise formulation of this property for pseudo-disks.

While for coloring pseudo-disks wrt. points there was no explicit upper bound, for the special case of homothets of a convex region[2], the constant was shown by Cardinal and Korman to be 4, just like for disks:

▶ **Theorem 11** ([4]). *Let P be the set of all points of the plane and $\mathcal{B}$ be a finite family of homothets of a given convex region, $\mathcal{I}(\mathcal{B}, P)$ admits a proper coloring with 4 colors.*

Recently it was proved by Keller and Smorodinsky that disks w.r.t. disks can be also colored in such a way, a common generalization of Theorems 7 and 9:

▶ **Theorem 12** ([11]). *Let $\mathcal{D}$ be the family of all disks in the plane and $\mathcal{B}$ a finite family of disks, $\mathcal{I}(\mathcal{B}, \mathcal{D})$ admits a proper coloring with 6 colors and a conflict-free coloring with $O(\log n)$ colors, where $n = |\mathcal{B}|$.*

While they did not prove the same for pseudo-disks, they solved two special cases (they stated these only as part of the proof of their main result):

▶ **Claim 13** ([11]). *Given a finite family $\mathcal{F}$ of pseudo-disks and a subfamily $\mathcal{B}$ of $\mathcal{F}$, If either $\mathcal{B}$ or $\mathcal{F} \setminus \mathcal{B}$ contains only pairwise disjoint pseudo-disks, then $\mathcal{I}(\mathcal{B}, \mathcal{F})$ admits a proper coloring with a constant number of colors and a conflict-free coloring with $O(\log n)$ colors, where $n = |\mathcal{B}|$.*

In this paper we generalize Theorem 12 to the case of coloring a pseudo-disk family wrt. another pseudo-disk family, which is a common generalization of all the above results (Theorems 7,9,10,11,12 and Claim 13). Moreover we prove the optimal upper bound of 4 colors, which improves the bound of Theorem 10 for coloring pseudo-disks wrt. disks from some constant number of colors to 4 colors and improves the bound of Theorem 12 for coloring disks wrt. disks from 6 colors to 4 colors. Furthermore, it provides an alternative proof for Theorems 9 and 11 (both of these were originally proved using dualization and solving equivalent problems about coloring points in the 3 dimensional space):

▶ **Theorem 14.** *Given a family $\mathcal{F}$ of pseudo-disks and a finite family $\mathcal{B}$ of pseudo-disks, $\mathcal{I}(\mathcal{B}, \mathcal{F})$ admits a proper coloring with 4 colors.*

Along the way we prove that the respective Delaunay-graph is planar:

▶ **Theorem 15.** *Given a family $\mathcal{F}$ of pseudo-disks and a finite family $\mathcal{B}$ of pseudo-disks, the Delaunay-graph of $\mathcal{B}$ wrt. $\mathcal{F}$ is a planar graph.*

This was not known even for the Delaunay-graph of pseudo-disks wrt. points. With standard methods (present also in the proof of Theorem 22) this already implies Theorem 14 with 6 colors instead of 4. To achieve the optimal bound we will need some additional ideas.

We mention that if $\mathcal{B}$ and $\mathcal{F}$ are both families of pairwise disjoint simply connected regions, then $\mathcal{I}(\mathcal{B}, \mathcal{F})$ is a planar hypergraph[3] and all planar hypergraphs can be generated this way. From this perspective Theorem 14 says that even with a much more relaxed definition of planarity of a hypergraph it remains 4-colorable.

Using the usual framework it easily follows from Theorem 14 that:

▶ **Corollary 16.** *Given a family $\mathcal{F}$ of pseudo-disks and a finite family $\mathcal{B}$ of pseudo-disks, $\mathcal{I}(\mathcal{B}, \mathcal{F})$ admits a conflict-free coloring with $O(\log n)$ colors, where $n = |\mathcal{B}|$.*

---

[2] For further results about homothets of a convex region see Section 4.

[3] The most common definition of a planar hypergraph is that its bipartite incidence graph is planar.

We note that Claim 13 implies in a straightforward way the main result of [11] about conflict-free coloring the (open/closed) neighborhood hypergraphs of intersection graphs of pseudo-disks (for definitions and details see [11]). Thus, Theorem 14 also implies their main result.

Note that in Theorem 14 $\mathcal{B}$ and $\mathcal{F}$ are not related in any way, thus among others, even though two convex regions can intersect infinitely many times, it implies somewhat surprisingly the following:

▶ **Corollary 17.** *We can proper color with* 4 *colors the family of homothets of a convex region A wrt. the family of homothets of another convex region B.*

1.2 Results related to families of linear union complexity

In [25] Theorem 10 was shown by proving a more general statement about coloring a family of regions that has linear union complexity wrt. points.

▶ **Definition 18.** Let $\mathcal{B}$ be a family of finitely many Jordan regions in the plane. The **vertices** of the **arrangement** of $\mathcal{B}$ are the intersection points of the boundaries of regions in $\mathcal{B}$, the **edges** are the maximal connected parts of the boundaries of regions in $\mathcal{B}$ that do not contain a vertex and the **faces** are the maximal connected parts of the plane which are disjoint from the edges and the vertices of the arrangement.

▶ **Definition 19.** The **union complexity** $\mathcal{U}(\mathcal{B})$ of a family of Jordan regions $\mathcal{B}$ is the number of edges of the arrangement $\mathcal{B}$ that lie on the boundary of $\cup_{B\in\mathcal{B}}B$.

We say that a family of regions $\mathcal{B}$ has **(c)-linear union complexity** if there exists a constant c such that for any subfamily $\mathcal{B}'$ of $\mathcal{B}$ the union complexity of $\mathcal{B}'$ is at most $c|\mathcal{B}'|$.[4]

▶ **Theorem 20** ([10])**.** *Any finite family of pseudo-disks in the plane has a linear union complexity.*

Theorem 20 shows that the following result of Smorodinsky is indeed more general than Theorem 10.

▶ **Theorem 21** ([25])**.** *Let P be the set of all points of the plane and $\mathcal{B}$ be a finite family of Jordan regions with linear union complexity. Then $\mathcal{I}(\mathcal{B},P)$ admits a proper coloring with a constant number of colors and a conflict-free coloring with $O(\log n)$ colors, where $n=|\mathcal{B}|$.*[5]

We generalize this in the following way:

▶ **Theorem 22.** *Given a family $\mathcal{F}$ of pseudo-disks and a finite family $\mathcal{B}$ of Jordan regions with linear union complexity, $\mathcal{I}(\mathcal{B},\mathcal{F})$ admits a proper coloring with constant number of colors.*

Note that using Theorem 20 we get that Theorem 22 implies Theorem 14 with a (non-explicit and worse) upper bound.

[4] Sometimes in the literature, in the definition of union complexity they count vertices rather than edges, yet it is easy to see that this does not affect the property of having linear union complexity. Also note that linear union complexity is not always defined hereditarily, we defined it this way in order to simplify our statements.

[5] When a statement is about a family with (c-)linear union complexity, by a constant we mean a constant that depends on c and the O notation similarly hides a dependence on c.

▶ **Corollary 23.** *Given a family $\mathcal{F}$ of pseudo-disks and a finite family $\mathcal{B}$ of Jordan regions with linear union complexity, $\mathcal{I}(\mathcal{B},\mathcal{F})$ admits a conflict-free coloring with $O(\log n)$ colors, where $n = |\mathcal{B}|$.*

The rest of the paper is structured as follows. In Section 2 we prove Theorem 15. In Section 3 we prove Theorem 14. Implications about conflict-free colorings and the results about regions of linear union complexity can be found in the full version of this paper. We also give an example that shows that we cannot always color properly points wrt. a family of linear union complexity with constantly many colors. This shows that Theorem 22 is strongest possible in the sense that we cannot change $\mathcal{F}$ from pseudo-disks to a family with linear union complexity, even if $\mathcal{B}$ would be only a finite family of points in the plane. Finally, in Section 4 we discuss some related (open) problems.

2 The Delaunay-graph of pseudo-disks wrt. pseudo-disks

In this section we prove Theorem 15. First we list some tools we need from the papers of Pinchasi and Buzaglo et al. about pseudo-disks:

▶ **Lemma 24** ([23])**.** *Let $\mathcal{F}$ be a family of pseudo-disks. Let $D \in \mathcal{F}$ and let $x \in D$ be any point. Then D can continuously be shrunk to the point x so that at each moment $\mathcal{F}$ is a family of pseudo-disks.*

Note that when shrinking this way, we can keep all shrunk copies of D in the family, it remains a pseudo-disk family as their boundaries are pairwise disjoint.

▶ **Definition 25.** We say that a pseudo-disk family $\mathcal{F}$ is **saturated** (for some family of regions $\mathcal{B}$) if we cannot add pseudo-disks to $\mathcal{F}$ in a way that strictly increases the number of hyperedges in $\mathcal{I}(\mathcal{B},\mathcal{F})$.

We can enlarge $\mathcal{F}$ greedily until it becomes saturated (we need to add at most $2^{|\mathcal{B}|}$ pseudo-disks).

▶ **Corollary 26.** *Given a family $\mathcal{F}$ of pseudo-disks and a finite family $\mathcal{B}$ of regions, there exists a saturated (for $\mathcal{B}$) family of pseudo-disks $\mathcal{F}'$ with $\mathcal{F}' \supseteq \mathcal{F}$ and consequently with $\mathcal{I}(\mathcal{B},\mathcal{F}') \supseteq \mathcal{I}(\mathcal{B},\mathcal{F})$. In particular, for every point p, $H_p = \{v_B : p \in B \in \mathcal{B}\}$ is a hyperedge of $\mathcal{I}(\mathcal{B},\mathcal{F}')$ (which we call the hyperedge that corresponds to p) if it is of size at least two.*

The second part of Corollary 26 holds as for a point p not contained in any member of $\mathcal{F}$ we can add to $\mathcal{F}$ a small disk centered at p while for a point p contained in some $F \in \mathcal{F}$ we can apply Lemma 24.

When proving our results, it will be enough to consider the case when $\mathcal{F}$ is saturated as adding hyperedges to a hypergraph cannot remove edges from its Delaunay-graph and cannot decrease the number of colors needed for a proper (or conflict-free) coloring. The following corollary of Lemma 24 will be useful when we deal with a saturated pseudo-disk family:

▶ **Corollary 27** ([23])**.** *Let $\mathcal{B}$ be a family of pairwise disjoint regions in the plane and let $\mathcal{F}$ be a family of pseudo-disks. Let D be a member of $\mathcal{F}$ and suppose that D intersects exactly k members of $\mathcal{B}$ one of which is the set $B \in \mathcal{B}$. Then for every $2 \le l \le k$ there exists a set $D' \subset D$ such that D' intersects B and exactly $l-1$ other regions from $\mathcal{B}$, and $\mathcal{F} \cup \{D'\}$ is again a family of pseudo-disks.*

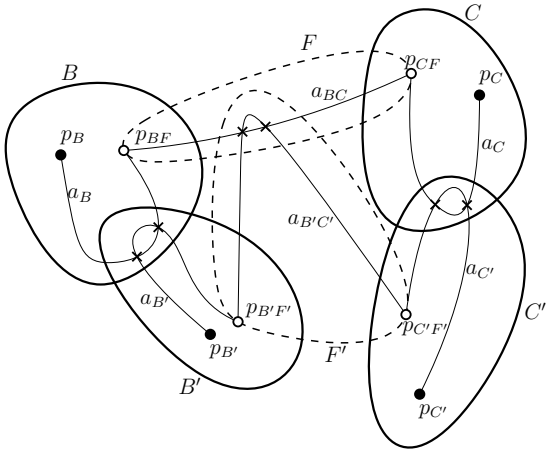


Figure 1 The edges e_f and $e_{f'}$ intersect an even number of times.

▶ **Lemma 28** ([3]). *Let D_1 and D_2 be two pseudo-disks in the plane. Let x and y be two points in $D_1 \setminus D_2$. Let a and b be two points in $D_2 \setminus D_1$. Let e be any Jordan arc connecting x and y that is fully contained in D_1. Let f be any Jordan arc connecting a and b that is fully contained in D_2. Then e and f cross an even number of times.*

▶ **Lemma 29** ([23]). *Given a family $\mathcal{F}$ of pseudo-disks and a finite family $\mathcal{B}$ of pairwise disjoint connected sets in the plane, the Delaunay-graph of $\mathcal{B}$ wrt. $\mathcal{F}$ is a planar graph.*

Note that for a family $\mathcal{B}$ of pairwise disjoint connected sets, the Delaunay-graph and the restricted Delaunay-graph are the same. Assuming also that $\mathcal{B}$ is a family of Jordan regions, Lemma 29 becomes a special case of Theorem 15. Before proving Theorem 15 we prove another special case which for Jordan regions strengthens Lemma 29, while its proof remains relatively simple:

▶ **Lemma 30.** *Given a family $\mathcal{F}$ of pseudo-disks and a finite family $\mathcal{B}$ of pseudo-disks such that each member $B \in B$ contains a point which is in no other $C \in \mathcal{B}$, the Delaunay-graph of $\mathcal{B}$ wrt. $\mathcal{F}$ is a planar graph.*

Proof. We will draw G in the plane in such a way that every pair of edges in G that do not share a common vertex cross an even number of times. The Hanani-Tutte Theorem [6, 29] then implies the planarity of G. In our case the edges may have self-crossings too but the Hanani-Tutte theorem holds in this case as well as we can easily redraw the edges to avoid self-crossings (for further details see, e.g., [24]).

For every $v_B, B \in \mathcal{B}$ choose a point $p_B \in B$ which is in no other $C \in \mathcal{B}$. For an illustration of the rest of the proof see Figure 1.

If v_B and v_C are connected by an edge f in G, then e_f, the drawing of the edge f, connecting p_B and p_C, is as follows. Let $F \in \mathcal{F}$ be a pseudo-disk corresponding to f. Draw an arc a_B inside B from p_B to a point $p_{BF} \in B \cap F$ ($p_{BF} \in C$ is allowed, also p_{BF} may coincide with p_B). Similarly draw an arc a_C inside C from p_C to a point $p_{CF} \in C \cap F$ ($p_{CF} \in B$ is allowed and p_{CF} may coincide with p_C). Finally, draw an arc a_{BC} inside F from p_{BF} to p_{CF} (p_{BF} and p_{CF} may also coincide in which case a_{BC} is of length zero). The concatenation of these three arcs is the drawing e_f of the edge f between the points p_B and p_C. Note that e_f may have self-crossings.

We are left to prove that in this drawing of G every pair of edges that do not share a vertex cross an even number of times.

Let $B, C, B', C' \in \mathcal{B}$ with edges f defined by F between v_B and v_C and f' defined by F' between $v_{B'}$ and $v_{C'}$ Suppose $e_f = a_B \cup a_{BC} \cup a_C$ and $e_{f'} = a_{B'} \cup a_{B'C'} \cup a_{C'}$ are the drawings of the two edges. Notice that the two endpoints of a_B are in $B \setminus B'$ and the two endpoints of $a_{B'}$ are in $B' \setminus B$. Thus using Lemma 28 we get that a_B with $a_{B'}$ intersects an even number of times. The same way we get that a_B with $a_{C'}$, a_C with $a_{B'}$ and a_C with $a_{C'}$ intersect an even number of times. As a_{BC} (resp. $a_{B'C'}$) is disjoint from B' and C' (resp. B and C), there is no intersection between a_{BC} and $a_{B'}, a_{C'}$ nor between $a_{B'C'}$ and a_B, a_C. Finally, the two endpoints of a_{BC} are in $F \setminus F'$ and the two endpoints of $a_{B'C'}$ are in $F' \setminus F$ thus again using Lemma 28 we get that a_{BC} and $a_{B'C'}$ intersect an even number of times. These together imply that indeed e_f and $e_{f'}$ intersect an even number of times, as required. ◀

Pach and Sharir [21] proved that (among others) pseudo-disks have linear union complexity using a similar approach as the proof of Lemma 30, connecting own-points of intersecting regions along their boundaries. More recently, Aronov et al. [2] proved (independently) the case of Lemma 30 when $\mathcal{B} = \mathcal{F}$ (with an almost identical proof) and showed that this implies that for a pseudo-disk family $\mathcal{F}$ and a finite subfamily $\mathcal{B}$ of $\mathcal{F}$, the intersection hypergraph $\mathcal{I}(\mathcal{B}, \mathcal{F})$ has VC-dimension at most 4. In fact, Lemma 30 implies the same way also that $\mathcal{I}(\mathcal{B}, \mathcal{F})$ has VC-dimension at most 4 when $\mathcal{B}$ is any finite pseudo-disk family, not necessarily a subfamily of $\mathcal{F}$, for details see the full version of this paper.

▶ **Corollary 31.** *Given a family $\mathcal{F}$ of pseudo-disks and a finite family $\mathcal{B}$ of pseudo-disks, the restricted Delaunay-graph of $\mathcal{B}$ wrt. $\mathcal{F}$ is a planar graph.*

The proof of Corollary 31 can be found in the full version of this paper.

▶ **Observation 32.** *If $\mathcal{F}$ is saturated (for $\mathcal{B}$), then for every $F \in \mathcal{F}$ the corresponding hyperedge $H = \{v_B : B \cap F \neq \emptyset\}$ of $\mathcal{I}(\mathcal{B}, \mathcal{F})$ either contains an edge of the restricted Delaunay-graph or F contains a point contained in at least 2 members of $\mathcal{B}$.*

Corollary 31 shows that Lemma 30 takes care of the planarity of the restricted Delaunay-graph. From Observation 32 we may think that we are left to take care of hyperedges that contain a point that is contained in at least 2 members of $\mathcal{B}$. This intuition turns out to be good, in all of our main results Lemma 30 will essentially reduce the problem to regarding the intersection hypergraph of $\mathcal{B}$ wrt. points (instead of $\mathcal{B}$ wrt. $\mathcal{F}$).

Before starting the proof of Theorem 15 we make some more preparations:

▶ **Definition 33.** Given a family of regions $\mathcal{B}$, a point is **k-deep** if it is contained in exactly k members of $\mathcal{B}$. We denote by ∂B the boundary of some region B of $\mathcal{B}$. We call a point which is in B but no other $C \in \mathcal{B}$, an **own-point** of B.

▶ **Definition 34.** A hypergraph $\mathcal{H}'$ **supports** another hypergraph $\mathcal{H}$ if they are on the same vertex set and for every hyperedge $H \in \mathcal{H}$ there exists a hyperedge $H' \in \mathcal{H}'$ such that $H' \subseteq H$.

▶ **Observation 35.** *If $\mathcal{H}''$ supports $\mathcal{H}'$ and $\mathcal{H}'$ supports $\mathcal{H}$, then $\mathcal{H}''$ supports $\mathcal{H}$.*

▶ **Observation 36.** *If a hypergraph $\mathcal{H}'$ supports another hypergraph $\mathcal{H}$, then the Delaunay-graph of $\mathcal{H}$ is a subgraph of the Delaunay-graph of $\mathcal{H}'$.*

Our last tool is the following theorem of Snoeyink and Hershberger (we write here a special case of what they called as the Sweeping theorem):

▶ **Theorem 37** ([27]). *Let Γ be a finite set of bi-infinite curves in the plane such that any pair of them intersects at most twice. Let d be a closed curve which intersects at most twice every curve in Γ. We can sweep d such that every member of the sweeping of d intersects at most twice every other curve of Γ.*

A sweeping of d in Theorem 37 is defined as a family $d(t)$, $t \in (-1, 1]$, of pairwise disjoint curves such that $d(0) = d$ and their union contains all points of the plane ($d(-1)$ is a degenerate curve consisting of a singular point).

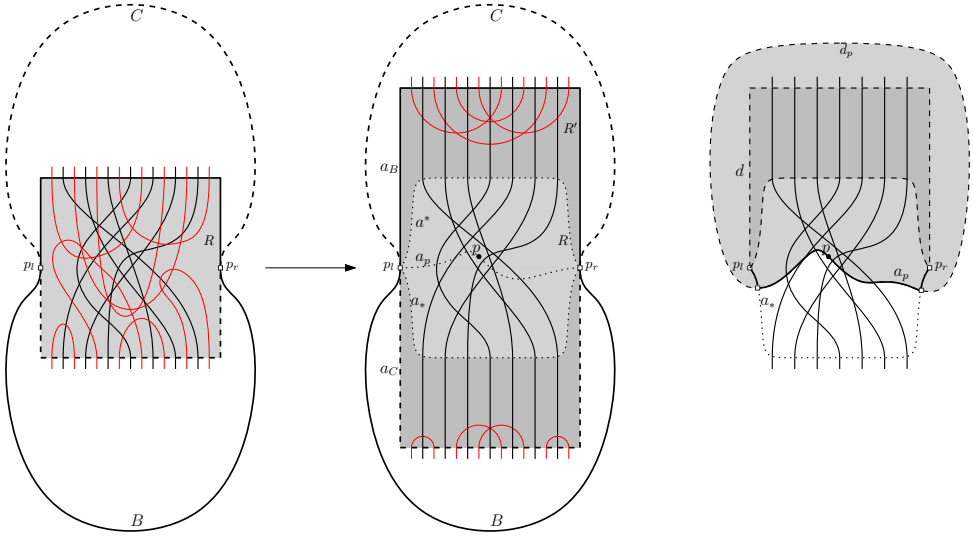
Proof of Theorem 15. Using Corollary 26 we can assume that $\mathcal{F}$ is saturated (for $\mathcal{B}$) and in particular, for every point p, $H_p = \{v_B : p \in B \in \mathcal{B}\}$ is a hyperedge of $\mathcal{I}(\mathcal{B}, \mathcal{F})$ if it is of size at least two.

We can keep deleting members of $\mathcal{B}$ from $\mathcal{B}$ which correspond to a vertex with degree 0 or 1 in the Delaunay-graph, during this process the Delaunay-graph of the new family keeps containing the graph induced by the original Delaunay-graph on the new (reduced) vertex set. If we can prove that the new Delaunay-graph is planar, then adding back the degree 0 and 1 vertices in reverse order we see that the original Delaunay-graph is also planar. Thus we can assume that every vertex of the Delaunay-graph has degree at least two. In this case for every $B \in \mathcal{B}$ we have a point in B which is in at most one other $C \in \mathcal{B}$ (as there are no degree-0 vertices) and there are no two regions $B, C \in \mathcal{B}$ such that $B \subset C$ (as there are no degree-0 or degree-1 vertices).

Given $\mathcal{B}$ and $\mathcal{F}$ we will modify $\mathcal{B}$ such that for the new family $\hat{\mathcal{B}}$ of pseudo-disks every $B \in \hat{\mathcal{B}}$ contains an own-point and furthermore we do this in a way that $\mathcal{I}(\hat{\mathcal{B}}, \mathcal{F})$ supports $\mathcal{I}(\mathcal{B}, \mathcal{F})$. Using Lemma 30 we get that the Delaunay-graph of $\hat{\mathcal{B}}$ wrt. $\mathcal{F}$ is planar and then by Observation 36 we get that the Delaunay graph of $\mathcal{B}$ wrt. $\mathcal{F}$ is also planar.

We do this modification by repeating the below defined operation finitely many times, at each time decreasing by at least one the number of members of $\mathcal{B}$ without an own-point. We now define the three steps of this operation.

- Step 1 - Preparation.
  Take an arbitrary $B \in \mathcal{B}$ that does not contain an own-point. We take a $C \in \mathcal{B}$ for which v_B is connected to v_C in the Delaunay-graph. Thus, there is a point p which is in $B \cap C$ but no other member of $\mathcal{B}$. We morph the plane such that $B \cap C$ becomes a square R such that the two intersection points of the boundary of B and C are on the horizontal halving line of the square (the upper part of ∂R belongs to ∂B and the lower part of ∂R belongs to ∂C) and no member of $\mathcal{B}$ (and $\mathcal{F}$) intersects the vertical sides of the square. This morphing is easily doable and it leaves intact the intersection structure of $\mathcal{B}$ and $\mathcal{F}$. Denote by p_l and p_r the intersection points of ∂B and ∂C, that is, the midpoints of the left and right side of R. This finishes the Preparation Step of our operation. See the left side of Figure 2a for what we get after this step.
- Step 2 - Pulling apart $B \cap C$.
  Now we define the Pulling apart $B \cap C$ Step of the operation which keeps the intersection structure of $\mathcal{B}$ and $\mathcal{F}$ outside $B \cap C$ intact. First we morph the plane such that the square's height is doubled to get the rectangle R' while its horizontal halving line does not change (the part of $\partial R'$ above the halving line is part of ∂B and the part below is part of ∂C) and the drawing inside R remains untouched. We do this such that the intersections of members of $\mathcal{B}$ (and of $\mathcal{F}$) with the horizontal sides of the square are stretched to become vertical lines in $R' \setminus R$. Next, for every $D \in \mathcal{B} \setminus \{B\}$ which intersects exactly one (horizontal) side of R', we redraw the part of ∂D inside B to be a half-circle inside B that has the same endpoints. See the right side of Figure 2a, where these redrawn



(a) Pulling apart $B \cap C$ **(b)** Drawing a_p.

Figure 2 Major steps of the operation used in the proof of Theorem 15.

boundary parts are drawn with thin red strokes. We get a modified family B', while the topology of $\mathcal{F}$ is unmodifed.

It is easy to see that we modified $\mathcal{B}$ in a way that $\mathcal{B}'$ remains a pseudo-disk family. Next we show that $\mathcal{I}(\mathcal{B}', \mathcal{F})$ supports $\mathcal{I}(\mathcal{B}, \mathcal{F})$, that is for every $H \in \mathcal{I}(\mathcal{B}, \mathcal{F})$ there exists a $H' \in \mathcal{I}(\mathcal{B}', \mathcal{F})$ such that $H' \subseteq H$. If $F \in \mathcal{F}$ contains only depth-1 points, then it is disjoint from $B \cap C$ and so the hyperedge H_F remains in the intersection hypergraph after pulling apart $B \cap C$. Otherwise, F contains a depth-2 point q and then H_F contains the hyperedge H_q. So it is enough to prove that for every H_q there is a hyperedge H' (also corresponding to some point) in $\mathcal{I}(\mathcal{B}', \mathcal{F})$ such that $H' \subseteq H_q$. For $q \notin B \cap C$, H_q remains in the intersection hypergraph after pulling apart $B \cap C$. Finally, for $q \in B \cap C$ the hyperedge H_q contains the hyperedge $\{v_B, v_C\}$ which is exactly the hyperedge corresponding to p in $\mathcal{B}'$ (recall that this is the point that was only in B and C and no other member of $\mathcal{B}$, and this property remains true for $\mathcal{B}'$ after the operation).

- Step 3 - Shrinking of C.

  Now we do the final step, the Shrinking of C Step of the operation. Let arc a_B (resp. a_C) be the part of $\partial R'$ above (resp. below) the halving line, which arc is also part of ∂B (resp. ∂C) after the first two steps of the operation. Let arc a^* (resp. arc a_*) be the part of ∂R which is above (resp. below) the halving line. We perturb the vertical parts of a^* and a_* slightly so that they do not overlap (nor intersect) with a_B and a_C. See right side of Figure 2a for an illustration.

  If either a^* or a_* contains a point p which is 2-deep (thus contained by B and C and no other member of $\mathcal{B}'$), then we redraw ∂C such that we change a_C to a^* or a_* (whichever contains the 2-deep point p) and then what we get remains a pseudo-disk family. This way the hyperedge $\{v_B, v_C\}$ is still in $\mathcal{I}(\mathcal{B}'', \mathcal{F})$ as it is corresponding to a point close to p. Thus $\mathcal{I}(\mathcal{B}'', \mathcal{F})$ supports $\mathcal{I}(\mathcal{B}', \mathcal{F})$, and so $\mathcal{I}(\mathcal{B}, \mathcal{F})$ as well. Furthermore, in $\mathcal{B}''$ there is a point close to p which is an own-point of B.

  Unfortunately it can happen that none of a^* and a_* contains a 2-deep point and so they are not suitable for redrawing a_C. Nevertheless, we assumed that there exists a 2-deep

point p in B before the operation, which remains 2-deep after the first two steps of the operation. In the following we find an arc a_p connecting p_l and p_r that goes inside R, goes through p and intersects exactly once every maximal part inside R of the boundary of a member of $\mathcal{B}'$, see Figure 2b. Assuming we have this arc a_p we can redraw ∂C such that we change a_C to a_p and what we get remains a pseudo-disk family. Also, the same way as in the previous case, we have that $\mathcal{I}(\mathcal{B}'', \mathcal{F})$ supports $\mathcal{I}(\mathcal{B}, \mathcal{F})$ and in $\mathcal{B}''$ there is a point close to p which is an own-point of B.

- Step 3b - Drawing a_p.
  While the existence of such an a_p is intuitively not surprising, proving its existence is a somewhat technical application of Theorem 37. The proof can be found in the full version of this paper.

Thus, with the above 3-step operation we changed $\mathcal{B}$ such that one more member, B, contains an own-point. Also it is easy to see that no other member could have lost its own-point, as we have redrawn members of $\mathcal{B}$ only inside $B \cap C$, where there were no 1-deep points. So after finitely many times repeating this operation we get a family $\hat{\mathcal{B}}$ in which every member has an own-point and whose intersection hypergraph $\mathcal{I}(\hat{\mathcal{B}}, \mathcal{F})$ supports the intersection hypergraph $\mathcal{I}(\mathcal{B}, \mathcal{F})$ we started with, concluding the proof. ◀

3 Coloring pseudo-disks wrt. pseudo-disks

Note that Observation 5 can be rephrased equivalently such that if the Delaunay-graph of $\mathcal{H}$ supports $\mathcal{H}$, then a proper coloring of the Delaunay-graph is a proper coloring of $\mathcal{H}$. More generally, it is true that:

▶ **Observation 38.** *If a hypergraph $\mathcal{H}'$ supports another hypergraph $\mathcal{H}$, then a proper coloring of $\mathcal{H}'$ is a proper coloring of $\mathcal{H}$.*

Proof of Theorem 14. Given the pseudo-disk family $\mathcal{F}$ and the finite pseudo-disk family $\mathcal{B}$, we want to color the vertices of $\mathcal{I}(\mathcal{B}, \mathcal{F})$ corresponding to the members of $\mathcal{B}$ such that for every $F \in \mathcal{F}$ the hyperedge H_F in $\mathcal{B}$ containing exactly those v_B for which $B \cap F \neq \emptyset$ is not monochromatic (assuming that $|H_F| \geq 2$). Using Corollary 26 and Observation 38 we can assume that $\mathcal{F}$ is saturated (for $\mathcal{B}$).

We prove the existence of a 4-coloring by induction first on the size of $\mathcal{B}$, second on the size of the largest containment-minimal hyperedges and third on the number of largest containment-minimal hyperedges. We say that a hypergraph $\mathcal{H}'$ is *better* than a hypergraph $\mathcal{H}$ (on the same vertex set) if either the size of the largest containment-minimal hyperedges is smaller in $\mathcal{H}'$ than in $\mathcal{H}$ or they are of the same size but there are more of them in $\mathcal{H}$. Notice that if $\mathcal{H}'$ supports $\mathcal{H}$, then $\mathcal{H}'$ is not worse than $\mathcal{H}$.

If $\mathcal{B}$ contains two pseudo-disks B, C such that $B \subset C$, then we can proper color $\mathcal{I}(\mathcal{B} \setminus \{B\}, \mathcal{F})$ by induction and then color v_B with a color different from the color of v_C to get a proper coloring of $\mathcal{I}(\mathcal{B}, \mathcal{F})$ as every hyperedge which contains v_B must contain v_C as well. Thus, we can assume that no two pseudo-disks in $\mathcal{B}$ contain one another.

We can assume that $\mathcal{F}$ is saturated (for $\mathcal{B}$) as adding hyperedges to the intersection hypergraph cannot make it worse. In particular, we can assume that $\mathcal{I}(\mathcal{B}, \mathcal{F})$ contains a hyperedge $H_p = \{v_B : p \in B \in \mathcal{B}\}$ (if it is of size at least two) for every point p in the plane.

Take one of the largest containment-minimal hyperedges, $H \in \mathcal{I}(\mathcal{B}, \mathcal{F})$.

If H is of size 2 that means that the Delaunay-graph supports $\mathcal{I}(\mathcal{B}, \mathcal{F})$ and then by Theorem 15 we can color it with 4 colors, which by Observation 38 is a proper coloring of $\mathcal{I}(\mathcal{B}, \mathcal{F})$, as required. This starts the induction.

Otherwise, H has $l \geq 3$ vertices. Assume by induction that every intersection hypergraph of pseudo-disks wrt. pseudo-disks better than $\mathcal{I}(\mathcal{B}, \mathcal{F})$ admits a proper 4-coloring.

If H corresponds to some F that contains only 1-deep points, then using Corollary 27 and that $\mathcal{F}$ is saturated we get that H is not containment-minimal, a contradiction. Thus, H contains a point which is at least 2-deep and then using that H is containment-minimal and $\mathcal{F}$ is saturated there must be a point p (any point inside a pseudo-disk corresponding to H is such) for which actually $H = H_p$. Take two members $B, C \in B$ for which $v_B, v_C \in H_p$. On $B \cap C$ we do the first two steps of the operation used in the proof of Theorem 15 to get a new pseudo-disk family $\mathcal{B}'$. As we have seen, the intersection hypergraph $\mathcal{I}(\mathcal{B}', \mathcal{F})$ supports $\mathcal{I}(\mathcal{B}, \mathcal{F})$ and thus $\mathcal{I}(\mathcal{B}', \mathcal{F})$ is not worse than $\mathcal{I}(\mathcal{B}, \mathcal{F})$. If $B \cap C$ in $\mathcal{B}'$ contains a 2-deep point, then $\{v_B, v_C\}$ is a hyperedge of $\mathcal{I}(\mathcal{B}', \mathcal{F})$ and the number of size-l hyperedges did actually decrease (as H is not containment minimal anymore) and so $\mathcal{I}(\mathcal{B}', \mathcal{F})$ is better than $\mathcal{I}(\mathcal{B}, \mathcal{F})$. Then by induction $\mathcal{I}(\mathcal{B}', \mathcal{F})$ can be colored with 4 colors, which by Observation 38 is a proper coloring of $\mathcal{I}(\mathcal{B}, \mathcal{F})$ as well.

The only case left is when $B \cap C$ still does not contain a 2-deep point. In this case similarly to Step 3 of the operation in the proof of Theorem 15, we redraw ∂C such that we change a_C to a^* (see again the right side of Figure 2a). In the new family $\mathcal{B}''$ every point in R is still covered at least twice, and nothing changed outside R, thus $\mathcal{I}(\mathcal{B}'', \mathcal{F})$ supports $\mathcal{I}(\mathcal{B}', F)$ (and then in turn it suppors $\mathcal{I}(\mathcal{B}, \mathcal{F})$). Moreover, the hyperedge corresponding to p is a subset of H_p but it does not contain v_C anymore, so it is a proper subset of H and is of size at least 2. This implies that the number of size-l hyperedges did decrease. Then again $\mathcal{I}(\mathcal{B}'', \mathcal{F})$ is better than $\mathcal{I}(\mathcal{B}, \mathcal{F})$ and then by induction can be colored with 4 colors, which by Observation 38 is a proper coloring of $\mathcal{I}(\mathcal{B}, \mathcal{F})$ as well. ◀

Observe that already for coloring points wrt. disks we may need 4 colors (as there are points whose intersection hypergraph wrt. disks is a complete graph on four vertices), so 4 is also a lower bound for coloring pseudo-disks wrt. pseudo-disks.

Without going into details, we note that from the proofs we can create (low-degree) polynomial time coloring algorithms, supposing that initially we are given reasonable amount of information (e.g., the whole structure of the arrangement (all the vertices, edges and faces) of the pseudo-disks of $\mathcal{B}$ and also for every hyperedge H in $\mathcal{I}(\mathcal{B}, \mathcal{F})$ we are given an F corresponding to H).

4 Discussion

As mentioned, the primary motivation behind considering such problems lies in applications to conflict-free colorings and to cover-decomposability problems.

When the main interest is in conflict-free colorings, the $\log n$ factor makes it less interesting to optimize the bound on the proper coloring result. However, when considering the relation to cover-decomposability problems, finding the exact value is of great interest. This makes it important that in Theorem 14 we could prove an exact upper bound for such a wide class of intersection hypergraphs. On the other hand, in Theorem 22 we aimed for keeping the proof as simple as possible, and so we did not optimize the number of colors.

In cover-decomposability problems the typical question asks if for some m we can properly 2-color $\mathcal{I}_m(\mathcal{B}, \mathcal{F})$, the subhypergraph of $\mathcal{I}(\mathcal{B}, \mathcal{F})$ containing only hyperedges of size at least m. If we properly color $\mathcal{I}_m(\mathcal{B}, \mathcal{F})$, then we say that we color $\mathcal{B}$ wrt. $\mathcal{F}$ for m.

Usually either $\mathcal{B}$ or $\mathcal{F}$ is the family of points of the plane. Originally researchers concentrated on the problem when the other family, $\mathcal{F}$ or $\mathcal{B}$, is the family of translates of a convex region (see, e.g., [20] for a history of this research direction).

In the past years researchers started to concentrate more on problems where $\mathcal{B}$ or $\mathcal{F}$ is the family of homothets of a convex region. One of the the most intriguing open problems is whether we can 2-color points wrt. homothets of a given convex polygon for some constant m. The existence of a 2-coloring for some m was proved for triangles [14] and the square [1] and recently for convex polygons the existence of a 3-coloring for some m was proved [17]. On the other hand, for coloring points wrt. disks 2 colors are not enough (for any m) [22]. As we have seen already in the introduction, a 4-coloring of points wrt. disks always exists (for $m = 2$), however the existence of a 3-coloring (for some m) is an open problem (see also the remark after Problem 39).

For the dual problem, about coloring the homothets of a convex polygon wrt. points, while proper 2-coloring exists for the homothets of triangles [14] for some m, for the homothets of other convex polygons there is not always a 2-coloring [18] (for any m). Also, for coloring disks wrt. points 2 colors are not enough (for any m) [19].

In [17] it was conjectured that points wrt. pseudo-disks and pseudo-disks wrt. points can be proper 3-colored (for some m). Encouraged by Theorem 14 we pose the common generalization of these conjectures:

▶ **Problem 39.** *Does there exist a constant m such that given a family $\mathcal{F}$ of pseudo-disks and a finite family $\mathcal{B}$ of pseudo-disks, $\mathcal{I}_m(\mathcal{B}, \mathcal{F})$ always admits a proper coloring with 3 colors?*

We note that Tóth [28] showed an example that choosing $m = 3$ is not enough already when $\mathcal{B}$ is a set of points and $\mathcal{F}$ is a family of pseudo-disks.

The case when $\mathcal{B}$ is a family of points and $\mathcal{F}$ is the family of disks was asked by the author of this paper 10 years ago [12, 13] and is still open (in this special case even $m = 3$ might be true).

As we mentioned in the introduction, one can easily change in the definition of the incidence hypergraph the incidence relation to the containment or reverse-containment relation. Thus we can define the following hypergraphs: $\mathcal{H}^{\subset}(\mathcal{B}, \mathcal{F})$ contains a hyperedge $H = \{v_B : B \subset F\}$ for every $F \in \mathcal{F}$; similarly, $\mathcal{H}^{\supset}(\mathcal{B}, \mathcal{F})$ contains a hyperedge $H = \{v_B : B \supset F\}$ for every $F \in \mathcal{F}$ (if these sets are of size at least two).

Thus we can ask if the respective variants of Theorem 14 and Problem 39 hold for these hypergraphs:

▶ **Problem 40.** *Given a family $\mathcal{F}$ of pseudo-disks and a finite family $\mathcal{B}$ of pseudo-disks, does $\mathcal{H}^{\subset}_m(\mathcal{B}, \mathcal{F})$ (resp. $\mathcal{H}^{\supset}_m(\mathcal{B}, \mathcal{F})$) always admit a proper coloring with 4 colors?*

Does there exist a constant m such that $\mathcal{H}^{\subset}_m(\mathcal{B}, \mathcal{F})$ (resp. $\mathcal{H}^{\supset}_m(\mathcal{B}, \mathcal{F})$) always admits a proper coloring with 3 colors?

The variant of this problem where $\mathcal{F}$ and $\mathcal{B}$ are families of intervals on the line was regarded in [16] where among others it was shown that for intervals these two classes (the class of containment and reverse-containment hypergraphs of intervals) are the same.

Finally we mention that similarly to pseudo-disks, pseudo-halfplanes are natural generalizations of halfplanes. For pseudo-halfplanes it was also possible to reprove the same coloring results that are known for halfplanes [15]. Also, due to the lack of direct relation to pseudo-disks, we did not mention axis-parallel rectangles, whose intersection hypergraph coloring problems are some of the most interesting open problems of the area.

References

1 Eyal Ackerman, Balázs Keszegh, and Mate Vizer. Coloring points with respect to squares. *Discrete & Computational Geometry*, jun 2017.

2 B. Aronov, A. Donakonda, E. Ezra, and R. Pinchasi. On Pseudo-disk Hypergraphs. *ArXiv e-prints*, 2018. arXiv:1802.08799.

3 Sarit Buzaglo, Rom Pinchasi, and Günter Rote. Topological hypergraphs. In *Thirty Essays on Geometric Graph Theory*, pages 71–81. Springer New York, oct 2012.

4 Jean Cardinal and Matias Korman. Coloring planar homothets and three-dimensional hypergraphs. *Computational geometry*, 46(9):1027–1035, 2013.

5 Timothy M. Chan and Sariel Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. *Discrete & Computational Geometry*, 48(2):373–392, 2012. doi:10.1007/s00454-012-9417-5.

6 Chaim Chojnacki. Über wesentlich unplättbare kurven im dreidimensionalen raume. *Fundamenta Mathematicae*, 23(1):135–142, 1934.

7 Guy Even, Zvi Lotker, Dana Ron, and Shakhar Smorodinsky. Conflict-free colorings of simple geometric regions with applications to frequency assignment in cellular networks. *SIAM Journal on Computing*, 33(1):94–136, jan 2003.

8 S. P. Fekete and P. Keldenich. Conflict-Free Coloring of Intersection Graphs. *ArXiv e-prints*, 2017. arXiv:1709.03876.

9 Andreas F. Holmsen, Hossein Nassajian Mojarrad, János Pach, and Gábor Tardos. Two extensions of the Erdős-Szekeres problem. *ArXiv e-prints*, 2017. arXiv:1710.11415.

10 Klara Kedem, Ron Livne, János Pach, and Micha Sharir. On the union of jordan regions and collision-free translational motion amidst polygonal obstacles. *Discrete & Computational Geometry*, 1(1):59–71, Mar 1986.

11 C. Keller and S. Smorodinsky. Conflict-Free Coloring of Intersection Graphs of Geometric Objects. *ArXiv e-prints*, 2017. arXiv:1704.02018.

12 Balázs Keszegh. Weak conflict-free colorings of point sets and simple regions. In Prosenjit Bose, editor, *Proceedings of the 19th Annual Canadian Conference on Computational Geometry, CCCG 2007, August 20-22, 2007, Carleton University, Ottawa, Canada*, pages 97–100. Carleton University, Ottawa, Canada, 2007. URL: http://cccg.ca/proceedings/2007/04b2.pdf.

13 Balázs Keszegh. Coloring half-planes and bottomless rectangles. *Computational geometry*, 45(9):495–507, 2012. doi:10.1016/j.comgeo.2011.09.004.

14 Balázs Keszegh and Dömötör Pálvölgyi. Octants are cover-decomposable. *Discrete & Computational Geometry*, 47(3):598–609, 2012.

15 Balázs Keszegh and Dömötör Pálvölgyi. An abstract approach to polychromatic coloring: Shallow hitting sets in aba-free hypergraphs and pseudohalfplanes. In Ernst W. Mayr, editor, *Graph-Theoretic Concepts in Computer Science - 41st International Workshop, WG 2015, Garching, Germany, June 17-19, 2015, Revised Papers*, volume 9224 of *Lecture Notes in Computer Science*, pages 266–280. Springer, 2015. doi:10.1007/978-3-662-53174-7_19.

16 Balázs Keszegh and Dömötör Pálvölgyi. More on decomposing coverings by octants. *Journal of Computational Geometry*, 6(1):300–315, 2015.

17 Balázs Keszegh and Dömötör Pálvölgyi. Proper coloring of geometric hypergraphs. In *Symposium on Computational Geometry*, volume 77 of *LIPIcs*, pages 47:1–47:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

18 István Kovács. Indecomposable coverings with homothetic polygons. *Discrete & Computational Geometry*, 53(4):817–824, 2015.

19 János Pach and Dömötör Pálvölgyi. Unsplittable coverings in the plane. In Ernst W. Mayr, editor, *Graph-Theoretic Concepts in Computer Science - 41st International Workshop, WG*

2015, Garching, Germany, June 17-19, 2015, Revised Papers, volume 9224 of *Lecture Notes in Computer Science*, pages 281–296. Springer, 2015. doi:10.1007/978-3-662-53174-7_20.

20 János Pach, Dömötör Pálvölgyi, and Géza Tóth. Survey on decomposition of multiple coverings. In *Geometry—Intuitive, Discrete, and Convex*, pages 219–257. Springer, 2013.

21 János Pach and Micha Sharir. On the boundary of the union of planar convex sets. *Discrete & Computational Geometry*, 21(3):321–328, 1999.

22 János Pach, Gábor Tardos, and Géza Tóth. Indecomposable coverings. In *Discrete Geometry, Combinatorics and Graph Theory*, pages 135–148. Springer, 2007.

23 Rom Pinchasi. A finite family of pseudodiscs must include a "small" pseudodisc. *SIAM Journal on Discrete Mathematics*, 28(4):1930–1934, jan 2014.

24 Fulek Radoslav, Michael J Pelsmajer, Marcus Schaefer, and Daniel Štefankovič. Adjacent crossings do matter. *Journal of Graph Algorithms and Applications*, 16(3):759–782, 2012.

25 Shakhar Smorodinsky. On the chromatic number of geometric hypergraphs. *SIAM Journal on Discrete Mathematics*, 21(3):676–687, 2007.

26 Shakhar Smorodinsky. Conflict-free coloring and its applications. In *Geometry—Intuitive, Discrete, and Convex*, pages 331–389. Springer, 2013.

27 Jack Snoeyink and John Hershberger. Sweeping arrangements of curves. In *Proceedings of the fifth annual symposium on Computational geometry*, pages 354–363. ACM, 1989.

28 Géza Tóth. personal communication.

29 William T Tutte. Toward a theory of crossing numbers. *Journal of Combinatorial Theory*, 8(1):45–53, 1970.

Minimizing Crossings in Constrained Two-Sided Circular Graph Layouts

Fabian Klute
Algorithms and Complexity Group, TU Wien
Vienna, Austria
fklute@ac.tuwien.ac.at

Martin Nöllenburg
Algorithms and Complexity Group, TU Wien
Vienna, Austria
noellenburg@ac.tuwien.ac.at
https://orcid.org/0000-0003-0454-3937

Abstract

Circular layouts are a popular graph drawing style, where vertices are placed on a circle and edges are drawn as straight chords. Crossing minimization in circular layouts is NP-hard. One way to allow for fewer crossings in practice are two-sided layouts that draw some edges as curves in the exterior of the circle. In fact, one- and two-sided circular layouts are equivalent to one-page and two-page book drawings, i.e., graph layouts with all vertices placed on a line (the *spine*) and edges drawn in one or two distinct half-planes (the *pages*) bounded by the spine. In this paper we study the problem of minimizing the crossings for a fixed cyclic vertex order by computing an optimal k-plane set of exteriorly drawn edges for $k \geq 1$, extending the previously studied case $k = 0$. We show that this relates to finding bounded-degree maximum-weight induced subgraphs of circle graphs, which is a graph-theoretic problem of independent interest. We show NP-hardness for arbitrary k, present an efficient algorithm for $k = 1$, and generalize it to an explicit XP-time algorithm for any fixed k. For the practically interesting case $k = 1$ we implemented our algorithm and present experimental results that confirm the applicability of our algorithm.

2012 ACM Subject Classification Human-centered computing → Graph drawings, Mathematics of computing → Graph algorithms, Theory of computation → Computational geometry

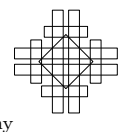
Keywords and phrases Graph Drawing, Circular Layouts, Crossing Minimization, Circle Graphs, Bounded-Degree Maximum-Weight Induced Subgraphs

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.53

Related Version A full version of this paper is available at `https://arxiv.org/abs/1803.05705`.

1 Introduction

Circular graph layouts are a popular drawing style to visualize graphs, e.g., in biology [16], and circular layout algorithms [21] are included in standard graph layout software [11] such as yFiles, Graphviz, or OGDF. In a circular graph layout all vertices are placed on a circle, while the edges are drawn as straight-line chords of that circle, see Fig. 1a. Minimizing the number of crossings between the edges is the main algorithmic problem for optimizing the readability of a circular graph layout. If the edges are drawn as chords, then all crossings are determined solely by the order of the vertices. By cutting the circle between any two vertices and straightening it, circular layouts immediately correspond to one-page book



drawings, in which all vertices are drawn on a line (the *spine*) and all edges are drawn in one half-plane (the *page*) bounded by the spine. Finding a vertex order that minimizes the crossings is NP-hard [18]. Heuristics and approximation algorithms have been studied in numerous papers, see, e.g., [2, 13, 20].

Gansner and Koren [8] presented an approach to compute improved circular layouts for a given input graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in a three-step process. The first step computes a vertex order of $\mathcal{V}$ that aims to minimize the overall edge length of the drawing, the second step determines a crossing-free subset of edges that are drawn outside the circle to reduce edge crossings in the interior (see Fig. 1b), and the third step introduces edge bundling to save ink and reduce clutter in the interior. The layouts by Gansner and Koren draw edges inside and outside the circle and thus are called *two-sided circular layouts*. Again, it is easy to see that two-sided circular layouts are equivalent to two-page book drawings, where the interior of the circle with its edges corresponds to the first page and the exterior to the second page.

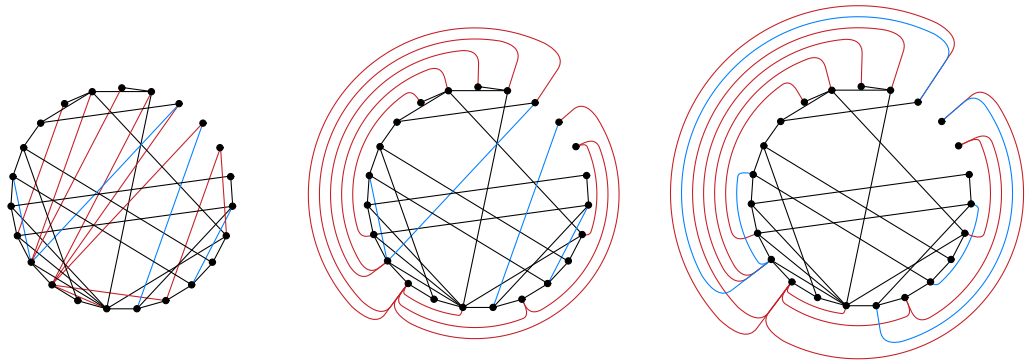
Inspired by their approach we take a closer look at the second step of the above process, which, in other words, determines for a given cyclic vertex order an outerplane subgraph to be drawn outside the circle such that the remaining crossings of the chords are minimized. Gansner and Koren [8] solve this problem in $O(|\mathcal{V}|^3)$ time.[1] In fact, the problem is equivalent to finding a maximum independent set in the corresponding circle graph $G^\circ = (V, E)$, which is the intersection graph of the chords (see Section 2 for details). The maximum independent set problem in a circle graph can be solved in $O(\ell)$ time [23], where ℓ is the total chord length of the circle graph (here $|\mathcal{E}| \leq \ell \leq |\mathcal{E}|^2$; see Section 4.2 for a precise definition of ℓ).

Contribution

We generalize the above crossing minimization problem from finding an outerplane graph to finding an outer k-plane graph, i.e., we ask for an edge set to be drawn outside the circle such that none of these edges has more than k crossings. Equivalently, we ask for a page assignment of the edges in a two-page book drawing, given a fixed vertex order, such that in one of the two pages each edge has at most k crossings. For $k = 0$ this is exactly the same problem considered by Gansner and Koren [8]. An example for $k = 1$ is shown in Fig. 1c. More generally, studying drawings of non-planar graphs with a bounded number of crossings per edge is a topic of great interest in graph drawing, see [15, 17].

We model the outer k-plane crossing minimization problem in two-sided circular layouts as a bounded-degree maximum-weight induced subgraph (BDMWIS) problem in the corresponding circle graph (Section 2). The BDMWIS problem is a natural generalization of the weighted independent set problem (setting the degree bound $k = 0$), which was the basis for Gansner and Koren's approach [8]. It is itself a weighted special case of the bounded-degree vertex deletion problem [3, 5, 7], a well-studied algorithmic graph problem of independent interest. For arbitrary k we show NP-hardness of the BDMWIS problem in Section 3. Our algorithms in Section 4 are based on dynamic programming using interval representations of circle graphs. For the case $k = 1$, where at most one crossing per exterior edge is permitted, we solve the BDMWIS problem for circle graphs in $O(|\mathcal{E}|^4)$ time. We then generalize our algorithm and obtain a problem-specific XP-time algorithm for circle graphs and any fixed k, whose running time is $O(|\mathcal{E}|^{2k+2})$. We note that the pure existence of an XP-time algorithm can also be derived from applying a metatheorem of Fomin et al. [6] using counting monadic second order (CMSO) logic, but the resulting running times are far worse. Finally, in Section 5, we present the results of a first experimental study comparing the crossing numbers of two-sided circular layouts for the cases $k = 0$ and $k = 1$.

[1] The paper claims $O(|\mathcal{V}|^2)$ time without a proof; the immediate running time of their algorithm is $O(|\mathcal{V}|^3)$.



(a) One-sided layout with 125 crossings.

(b) Two-sided layout for $k = 0$ with 48 crossings.

(c) Two-sided layout for $k = 1$ with 30 crossings.

Figure 1 Circular layouts of a graph $(\mathcal{G}, \pi)$ (23 vertices, 45 edges) computed by our algorithms.

2 Preliminaries

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph and π a cyclic order of $\mathcal{V}$. We arrange the vertices in order π on a circle C and draw edges as straight chords to obtain a (*one-sided*) *circular drawing* Γ, see Fig. 1a. Note that all crossings of Γ are fully determined by π: two edges cross iff their endpoints alternate in π. Our goal in this paper is to find a subset of edges to be drawn in the unbounded region outside C with no more than k crossings per edge in order to minimize the total number of edge crossings or the number of remaining edge crossings inside C.

More precisely, in a *two-sided circular drawing* Δ of $(\mathcal{G}, \pi)$ we still draw all vertices on a circle C in the order π, but we split the edges into two disjoint sets $\mathcal{E}_1$ and $\mathcal{E}_2$ with $\mathcal{E}_1 \cup \mathcal{E}_2 = \mathcal{E}$. The edges in $\mathcal{E}_1$ are drawn as straight chords, while the edges in $\mathcal{E}_2$ are drawn as simple curves in the exterior of C, see Fig. 1c. Asking for a set $\mathcal{E}_2$ that globally minimizes the crossings in Δ is equivalent to the NP-hard fixed linear crossing minimization problem in 2-page book drawings [19]. Hence we add the additional constraint that the exterior drawing induced by $\mathcal{E}_2$ is *outer k-plane*, i.e., each edge in $\mathcal{E}_2$ is crossed by at most k other edges in $\mathcal{E}_2$. This is motivated by the fact that, due to their detour, exterior edges are already harder to read and hence should not be further impaired by too many crossings. The parameter k, which can be assumed to be small, gives us control on the maximum number of crossings per exterior edge. Previous work [8] is limited to the case $k = 0$.

2.1 Problem transformation

Instead of working with a one-sided input layout Γ of $(\mathcal{G}, \pi)$ directly we consider the corresponding *circle graph* $G^\circ = (V, E)$ of $(\mathcal{G}, \pi)$. The vertex set V of G° has one vertex for each edge in $\mathcal{E}$ and two vertices $u, v \in V$ are connected by an edge (u, v) in E if and only if the chords corresponding to u and v cross in Γ, i.e., their endpoints alternate in π. The number of vertices $|V|$ of G° thus equals the number of edges $|\mathcal{E}|$ of $\mathcal{G}$ and the number of edges $|E|$ of G° equals the number of crossings in Γ. Moreover, the degree $\deg(v)$ of a vertex v in G° is the number of crossings of the corresponding edge in Γ.

Next we show that we can reduce our outer k-plane crossing minimization problem in two-sided circular layouts of $(\mathcal{G}, \pi)$ to an instance of the following bounded-degree maximum-weight induced subgraph problem for G°.

▶ **Problem 1** (Bounded-Degree k Maximum-Weight Induced Subgraph (k-BDMWIS)). *Let $G = (V, E)$ be a weighted graph with a vertex weight $w(v) \in \mathbb{R}^+$ for each $v \in V$ and an edge weight $w(u, v) \in \mathbb{R}^+$ for each $(u, v) \in E$ and let $k \in \mathbb{N}$. Find a set $V' \subset V$ such that the induced subgraph $G[V'] = (V', E')$ has maximum vertex degree k and maximizes the weight*

$$W = W(G[V']) = \sum_{v \in V'} w(v) - \sum_{(u,v) \in E'} w(u, v).$$

For general graphs it follows immediately from Yannakakis [24] that k-BDMWIS is NP-hard, but restricting the graph class to circle graphs makes the problem significantly easier, at least for constant k, as we show in this paper.

For our reduction it remains to assign suitable vertex and edge weights to G°. We define $w(v) = \deg(v)$ for all vertices $v \in V$ and $w(u, v) = 1$ or, alternatively, $w(u, v) = 2$ for all edges $(u, v) \in E$, depending on the type of crossings to minimize.

▶ **Lemma 2.** *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph with cyclic vertex order π and $k \in \mathbb{N}$. Then a maximum-weight degree-k induced subgraph in the corresponding weighted circle graph $G^\circ = (V, E)$ induces an outer k-plane graph that minimizes the number of crossings in the corresponding two-sided layout Δ of $(\mathcal{G}, \pi)$.*

Proof. Let $V^* \subset V$ be a vertex set that induces a maximum-weight subgraph of degree at most k in G°. Since vertices in G° correspond to edges in $\mathcal{G}$, we can choose $\mathcal{E}^* = V^*$ as the set of exterior edges in Δ. Each edge in G° corresponds to a crossing in the one-sided circular layout Γ. Hence each edge in the induced graph $G^\circ[V^*]$ corresponds to an exterior crossing in Δ. Since the maximum degree of $G^\circ[V^*]$ is k, no exterior edge in Δ has more than k crossings.

The degree of a vertex $v \in V^*$ (and thus its weight $w(v)$) equals the number of crossings that are removed from Γ by drawing the corresponding edge in $\mathcal{E}^*$ in the exterior part of Δ. However, if two vertices in V^* are connected by an edge, their corresponding edges in $\mathcal{E}^*$ necessarily cross in the exterior part of Δ and we need to add a correction term, otherwise the crossing would be counted twice. So for edge weights $w(u, v) = 1$ the weight W maximized by V^* equals the number of crossings that are removed from the interior part of Δ. For $w(u, v) = 2$, though, the weight W equals the number of crossings that are removed from the interior, but not counting those that are simply shifted to the exterior of Δ. ◀

Lemma 2 tells us that instead of minimizing the crossings in two-sided circular layouts with an outer k-plane exterior graph, we can focus on solving the k-BDMWIS problem for circle graphs in the rest of the paper.

2.2 Interval representation of circle graphs

There are two alternative representations of circle graphs. The first one is the *chord representation* as a set of chords of a circle (i.e., a one-sided circular layout), whose intersection graph actually serves as the very definition of circle graphs. The second and less immediate representation is the *interval representation* as an *overlap graph*, which is more convenient for describing our algorithms. In an interval representation each vertex is represented as a closed interval $I \subset \mathbb{R}$. Two vertices are adjacent if and only if the two corresponding intervals partially overlap, i.e., they intersect but neither interval contains the other.

Gavril [10] showed that circle graphs and overlap graphs represent the same class of graphs. To obtain an interval representation from a chord representation Γ on a circle C the idea is to pick a point p on C, which is not the endpoint of a chord, rotate Γ such that

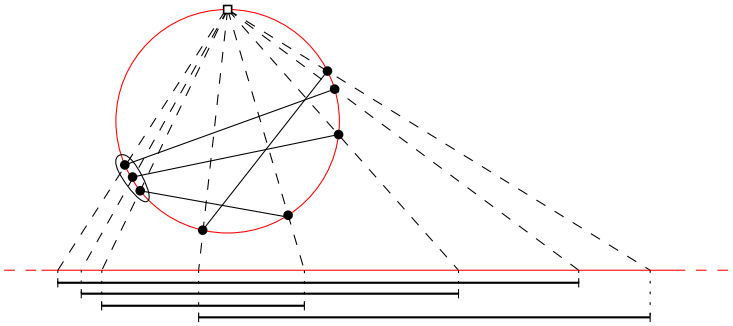


Figure 2 An example projection of the chord representation of a circle graph (here: $K_{1,3}$) to obtain an interval representation of the same graph as an overlap graph. Marked groups of endpoints indicate how chords incident to the same vertex are separated before the projection.

p is the topmost point of Γ and project the chords from p onto the real line below C, see Fig. 2. Each chord is then represented as a finite interval and two chords intersect if and only if their projected intervals partially overlap. We can further assume that all endpoints of the projected intervals are distinct by locally separating chords with a shared endpoint in Γ before the projection, such that the intersection graph of the chords does not change.

3 NP-hardness

For arbitrary, non-constant $k \in \mathbb{N}$ we show that k-BDMWIS is NP-hard, even on circle graphs. Our reduction is from the Minimum Dominating Set problem, which is NP-hard on circle graphs [12].

▶ **Problem 3** (Minimum Dominating Set). *Given a graph $G = (V, E)$, find a set $V' \subseteq V$ of minimum cardinality such that for each $u \in V \setminus V'$ there is a vertex $v \in V'$ with $(u, v) \in E$.*

▶ **Theorem 4.** *k-BDMWIS is NP-hard on circle graphs, even if all vertex weights are one and all edge weights are zero.*

Proof. Given an instance of Minimum Dominating Set on a circle graph $G = (V, E)$ we construct an instance of k-BDMWIS. First let $G' = (V', E')$ be a copy of G. We set the degree bound k equal to the maximum degree of G and attach new leaves to each vertex $v \in V'$ until every (non-leaf) vertex in G' has degree $k + 1$. Note that G' remains a circle graph when adding leaves. We set the weights to $w(v) = 1$ for $v \in V'$ and $w(u, v) = 0$ for $(u, v) \in E'$. This implies that the weight W to be maximized is just the number of vertices in the induced subgraph.

Now given a minimum dominating set $V_d \subseteq V$ of G, we know that for every vertex $v \in V$ either $v \in V_d$ or there exists a vertex $u \in V_d$ such that $(u, v) \in E$. This means if we set $V_s = V' \setminus V_d$ the graph $G'[V_s]$ has maximum degree k, since for every $v \in V_s$ at least one neighbor is in V_d and the maximum degree in G' is $k+1$. Since V_d is a minimum dominating set, V_s, for which we can assume that it contains all leaves, is the largest set of vertices such that $G'[V_s]$ has maximum degree k. Hence V_s is a solution to the k-BDMWIS problem on G'.

Conversely let $V_s \subseteq V'$ be a solution to the k-BDMWIS problem on G'. Again we can assume that V_s contains all leaves of G'. Otherwise let $u \in V' \setminus V_s$ be a leaf of G' with unique neighbor $v \in V'$. The only possible reason that $u \notin V_s$ is that $v \in V_s$ and the degree of v in $G'[V_s]$ is $\deg(v) = k$. If we replace in V_s a non-leaf neighbor w of v (which must exist) by the

leaf u, the resulting set has the same cardinality and satisfies the degree constraint. Now let $V_d = V' \setminus V_s$. By our assumption V_d contains no leaves of G' and $V_d \subseteq V$. Since every vertex in $G'[V_s]$ has degree at most k we know that each $v \in V \setminus V_d$ must have a neighbor $u \in V_d$, otherwise it would have degree $k+1$ in $G'[V_s]$. Thus V_d is a dominating set. Further, V_d is a minimum dominating set. If there was a smaller dominating set V_d' in G then $V' \setminus V_d'$ would be a larger solution than V_s for the k-BDMWIS problem on G', which is a contradiction. ◀

4 Algorithms for k-BDMWIS on circle graphs

Before describing our dynamic programming algorithms for $k = 1$ and the generalization to $k \geq 2$ in this section, we introduce the necessary basic definitions and notation using the interval perspective on k-BDMWIS for circle graphs.

4.1 Notation and definitions

Let $G = (V, E)$ be a circle graph and $\mathcal{I} = \{I_1, \ldots, I_n\}$ an interval representation of G with n intervals that have $2n$ distinct endpoints as defined in Section 2.2. Let $\sigma(\mathcal{I}) = \{\sigma_1, \ldots, \sigma_{2n}\}$ be the set of all interval endpoints and assume that they are sorted in increasing order, i.e., $\sigma_i < \sigma_j$ for all $i < j$. We may in fact assume without loss of generality that $\sigma(\mathcal{I}) = \{1, \ldots, 2n\}$ by mapping each interval $[\sigma_l, \sigma_r] \in \mathcal{I}$ to the interval $[l, r]$ defined by its index ranks. Clearly the order of the endpoints of two intervals $[\sigma_l, \sigma_r]$ and $[\sigma_{l'}, \sigma_{r'}]$ is exactly the same as the order of the endpoint of the intervals $[l, r]$ and $[l', r']$ and thus the overlap or circle graph defined by the new interval set is exactly the same as the one defined by $\mathcal{I}$.

For two distinct intervals $I = [a, b]$ and $J = [c, d] \in \mathcal{I}$ we say that I and J *overlap* if $a < c < b < d$ or $c < a < d < b$. Two overlapping intervals correspond to an edge in G. For an interval $I \in \mathcal{I}$ and a subset $\mathcal{I}' \subseteq \mathcal{I}$ we define the overlap set $\mathcal{P}(I, \mathcal{I}') = \{J \mid J \in \mathcal{I}' \text{ and } I, J \text{ overlap}\}$. Further, for $I = [a, b]$, we define the forward overlap set $\overrightarrow{\mathcal{P}}(I, \mathcal{I}') = \{J \mid J = [c, d] \in \mathcal{P}(I, \mathcal{I}') \text{ and } c < b < d\}$ of intervals overlapping on the right side of I and the set $\mathcal{P}(\mathcal{I}') = \{\{I, J\} \mid I, J \in \mathcal{I}' \text{ and } J \in \mathcal{P}(I, \mathcal{I}')\}$ of all overlapping pairs of intervals in $\mathcal{I}'$. If $J \subset I$, i.e., $a < c < d < b$, we say that I *nests* J (or J is *nested* in I). Nested intervals do not correspond to edges in G. For a subset $\mathcal{I}' \subseteq \mathcal{I}$ we define the set of all intervals nested in I as $\mathcal{N}(I, \mathcal{I}') = \{J \mid J \in \mathcal{I}' \text{ and } J \text{ is nested in } I\}$.

Let $\mathcal{I}' \subseteq \mathcal{I}$ be a set of n' intervals. We say $\mathcal{I}'$ is *connected* if its corresponding overlap or circle graph is connected. Further let $\sigma(\mathcal{I}') = \{i_1, \ldots, i_{2n'}\}$ be the sorted interval endpoints of $\mathcal{I}'$. The *span* of $\mathcal{I}'$ is defined as $\text{span}(\mathcal{I}') = i_{2n'} - i_1$ and the *fit* of the set $\mathcal{I}'$ is defined as $\text{fit}(\mathcal{I}') = \max\{i_{j+1} - i_j \mid 1 \leq j < 2n'\}$.

For a weighted circle graph $G = (V, E)$ with interval representation $\mathcal{I}$ we can immediately assign each vertex weight $w(v)$ as an interval weight $w(I_v)$ to the interval $I_v \in \mathcal{I}$ representing v and each edge weight $w(u, v)$ to the overlapping pair of intervals $\{I_u, I_v\} \in \mathcal{P}(\mathcal{I})$ that represents the edge $(u, v) \in E$. We can now phrase the k-BDMWIS problem for a circle graph in terms of its interval representation, i.e., given an interval representation $\mathcal{I}$ of a circle graph G, find a subset $\mathcal{I}' \subseteq \mathcal{I}$ such that no $I \in \mathcal{I}'$ overlaps more than k other intervals in $\mathcal{I}'$ and such that the weight $W(\mathcal{I}') = \sum_{I \in \mathcal{I}'} w(I) - \sum_{\{I,J\} \in \mathcal{P}(\mathcal{I}')} w(I, J)$ is maximized. We call such an optimal subset $\mathcal{I}'$ of intervals a *max-weight k-overlap set*.

4.2 Properties of max-weight 1-overlap sets

The basic idea for our dynamic programming algorithm is to decompose any 1-*overlap set*, i.e., a set of intervals, in which no interval overlaps more than one other interval, into a sequence of independent single intervals and overlapping interval pairs. Consequently, we can

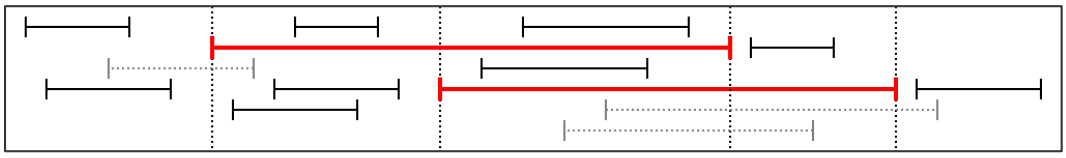


Figure 3 Split along the two thick red intervals. The dotted intervals are discarded and we recurse on the five sets with black intervals.

find a max-weight 1-overlap set by optimizing over all possible ways to select a single interval or an overlapping interval pair and recursively solving the induced independent subinstances that are obtained by splitting the instance according to the selected interval(s).

Let $\mathcal{I}$ be a set of intervals. For $x, y \in \mathbb{R} \cup \{\pm\infty\}$ with $x \leq y$ we define the set $\mathcal{I}[x, y] = \{I \in \mathcal{I} \mid I \subseteq [x, y]\}$ as the subset of $\mathcal{I}$ contained in $[x, y]$. Note that $\mathcal{I}[-\infty, \infty] = \mathcal{I}$. For any $I = [a, b] \in \mathcal{I}[x, y]$ we can *split* $\mathcal{I}[x, y]$ *along* I into the three sets $\mathcal{I}[x, a]$, $\mathcal{I}[a, b]$, $\mathcal{I}[b, y]$. This split corresponds to selecting I as an interval without overlaps in a candidate 1-overlap set. All intervals which are not contained in one of the three sets will be discarded after the split.

Similarly, we can split $\mathcal{I}[x, y]$ along a pair of overlapping intervals $I = [a, b], J = [c, d] \in \mathcal{I}$ to be included in candidate solution. Without loss of generality let $a < c < b < d$. Then the split creates the five sets $\mathcal{I}[x, a]$, $\mathcal{I}[a, c]$, $\mathcal{I}[c, b]$, $\mathcal{I}[b, d]$, $\mathcal{I}[d, y]$, see Fig. 3. Again, all intervals which are not contained in one of the five sets are discarded. The next lemma shows that none of the discarded overlapping intervals can be included in a 1-overlap set together with I and J.

▶ **Lemma 5.** *For any $x \in \mathbb{R}$ at most two overlapping intervals $I = [a, b], J = [c, d] \in \mathcal{I}$ with $a \leq x \leq b$ and $c \leq x \leq d$ can be part of a 1-overlap set of $\mathcal{I}$.*

Proof. Assume there is a third interval $K = [e, f] \in \mathcal{I}$ with $e \leq x \leq f$ in a 1-overlap set, which overlaps I or J or both. Interval K cannot be added to the 1-overlap set without creating at least one interval that overlaps two other intervals, which is not allowed in a 1-overlap set. ◀

Our algorithm for the max-weight 1-overlap set problem extends some of the ideas of the algorithm presented by Valiente for the independent set problem in circle graphs [23]. In our analysis we use Valiente's notion of *total chord length*, where the chord length is the same as the *length* $\ell(I) = j - i$ of the corresponding interval $I = [i, j] \in \mathcal{I}$. The *total interval length* can then be defined as $\ell = \ell(\mathcal{I}) = \sum_{I \in \mathcal{I}} \ell(I)$. We use the following bound in our analysis.

▶ **Lemma 6.** *Let $\mathcal{I}$ be a set of intervals and γ be the maximum degree of the corresponding overlap or circle graph, then $\sum_{I \in \mathcal{I}} \sum_{J \in \mathcal{P}(I, \mathcal{I})} (\ell(I) + \ell(J)) = O(\gamma \ell)$.*

Proof. We first observe that $J \in \mathcal{P}(I, \mathcal{I})$ if and only if $I \in \mathcal{P}(J, \mathcal{I})$. So in total each interval in $\mathcal{I}$ appears at most γ times as I and at most γ times as J in the double sum, i.e., no interval in $\mathcal{I}$ appears more than 2γ times and the bound follows. ◀

4.3 An algorithm for max-weight 1-overlap sets

Our algorithm to compute max-weight 1-overlap sets runs in two phases. In the first phase, we compute the weights of optimal solutions on subinstances of increasing size by recursively re-using solutions of smaller subinstances. In the second phase we optimize over all ways of combining optimal subsolutions to obtain a max-weight 1-overlap set.

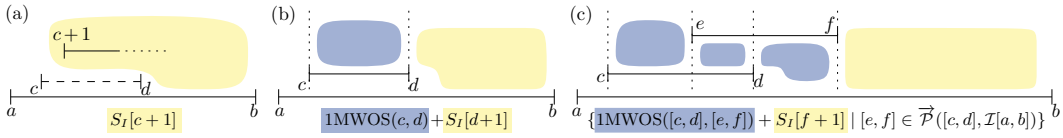


Figure 4 Illustration of Recurrence (3). The dashed intervals are discarded, while solid ones are considered in the solution.

The subinstances of interest are defined as follows. Let $\mathcal{I}' \subseteq \mathcal{I}$ be a connected set of intervals and let $l = l(\mathcal{I}')$ and $r = r(\mathcal{I}')$ be the leftmost and rightmost endpoints of all intervals in $\mathcal{I}'$. We define the value $\text{1MWOS}(\mathcal{I}')$ as the maximum weight of a 1-overlap set on $\mathcal{I}[l, r]$ that includes $\mathcal{I}'$ in the 1-overlap set (if one exists). Lemma 5 implies that it is sufficient to compute the 1MWOS values for single intervals $I \in \mathcal{I}$ and overlapping pairs $I, J \in \mathcal{I}$ since any connected set of three or more intervals cannot be a 1-overlap set any more.

We start with the computation of $\text{1MWOS}(I)$ for a single interval $I = [a, b] \in \mathcal{I}$. Using a recursive computation scheme of 1MWOS that uses increasing interval lengths we may assume by induction that for any interval $J \in \mathcal{I}$ with $\ell(J) < \ell(I)$ and any overlapping pair of intervals $J, K \in \mathcal{I}$ with $\text{span}(J, K) < \ell(I)$ the sets $\text{1MWOS}(J)$ and $\text{1MWOS}(J, K)$ are already computed. If we select I for the 1-overlap set as a single interval without overlaps, we need to consider for $\text{1MWOS}(I)$ only those intervals nested in I. Refer to Fig. 4 for an illustration. The value of $\text{1MWOS}(I)$ is determined using an auxiliary recurrence $S_I[x]$ for $a \leq x \leq b$ and the weight $w(I)$ resulting from the choice of I:

$$\text{1MWOS}([a, b]) = S_I[a + 1] + w(I). \tag{1}$$

For a fixed interval $I = [a, b]$ the value $S_I[x]$ represents the weight of an optimal solution of $\mathcal{I}[x, b]$. To simplify the definition of recurrence $S_I[x]$ we define the set $D_S([c, d], \mathcal{I}[a, b])$ with $[c, d] \in \mathcal{I}[a, b]$, in which we collect all 1MWOS values for pairs composed of $[c, d]$ and an interval in $\overrightarrow{\mathcal{P}}([c, d], \mathcal{I}[a, b])$ (see Fig. 4(c)) as

$$D_S([c, d], \mathcal{I}[a, b]) = \{\text{1MWOS}([c, d], [e, f]) + S_I[f + 1] \mid [e, f] \in \overrightarrow{\mathcal{P}}([c, d], \mathcal{I}[a, b])\}. \tag{2}$$

The main idea of the definition of $S_I[x]$ is a maximization step over the already computed sub-solutions that may be composed to an optimal solution for $\mathcal{I}[x, b]$. To stop the recurrence we set $S_I[b] = 0$ and for every end-point d of an interval $[c, d] \in \mathcal{I}[a, b]$ we set $S_I[d] = S_I[d+1]$. It remains to define the recurrence for the start-point c of each interval $[c, d] \in \mathcal{I}[a, b]$:

$$S_I[c] = \max\{\{S_I[c + 1], \text{1MWOS}([c, d]) + S_I[d + 1]\} \cup D_S([c, d], \mathcal{I}[a, b])\}. \tag{3}$$

Figure 4 depicts which of the possible configurations of selected intervals is represented by which values in the maximization step of eq. (3). The first option (Fig. 4(a)) is to discard the interval $[c, d]$, the second option (Fig. 4(b)) is to select $[c, d]$ as a single interval, and the third option (Fig. 4(c)) is to select $[c, d]$ and an interval in its forward overlap set.

▶ **Lemma 7.** *Let $\mathcal{I}$ be a set of intervals and $I \in \mathcal{I}$, then the value $1MWOS(I)$ can be computed in $O(\gamma\, \ell(I))$ time assuming all $1MWOS(J)$ and $1MWOS(J, K)$ values are computed for $J, K \in \mathcal{I}$, $\ell(J) < \ell(I)$ and $\text{span}(J, K) < \ell(I)$.*

Proof. Recurrence (1) is correct if $S[a+1]$ is exactly the weight of a max-weight 1-overlap set on the set $\mathcal{N}(I, \mathcal{I})$, the set of nested intervals of I. The proof is by induction over the number of intervals in $\mathcal{N}(I, \mathcal{I})$. In case $\mathcal{N}(I, \mathcal{I})$ is empty $b = a + 1$ and with $S[b] = 0$ Recurrence (1) is correct.

Now let $\mathcal{N}(I,\mathcal{I})$ consist of one or more intervals. By Lemma 5 there can only be three cases of how an interval $J \in \mathcal{N}(I,\mathcal{I})$ contributes. We can decide to discard J, to add it as a singleton interval which allows us to split $\mathcal{N}(I,\mathcal{I})$ along J or to add an overlapping pair $J,K \in \mathcal{N}(I,\mathcal{I})$ such that $K \in \overrightarrow{\mathcal{P}}(J,\mathcal{I}[a,b])$ and split $\mathcal{N}(I,\mathcal{I})$ along J,K.

For the start-point c of an interval $J=[c,d]$ the maximization in the definition of S in Recurrence (3) exactly considers these three possibilities (recall Fig. 4). For all end-points aside from b we simply use the value of the next start-point or $S[b]=0$ which ends the recurrence. Since all 1MWOS(J) and 1MWOS(J,K) are computed for $J,K \in \mathcal{I}$, $\ell(J)<\ell(I)$ and $\operatorname{span}(J,K)<\ell(I)$ the auxiliary table S is computed in one iteration across $\sigma(\mathcal{I}[a,b])$.

The overall running time is dominated by traversing the D_S sets, which contain at most γ values. This has to be done for every start-point of an interval in $\mathcal{N}(I,\mathcal{I})$ which leads to an overall computation time of $O(\gamma\,\ell(I))$ for 1MWOS(I). ◀

Until now we only considered computing the 1MWOS value of a single interval, but we still need to compute 1MWOS for pairs of overlapping intervals. Let $I=[c,d], J=[e,f] \in \mathcal{I}$ be two intervals such that $J \in \overrightarrow{\mathcal{P}}(I,\mathcal{I})$. If we split $\mathcal{I}$ along these two intervals we find three independent regions (recall Fig. 4(c)) and obtain

$$\text{1MWOS}(I,J) = L_{I,J}[c+1] + M_{I,J}[e+1] + R_{I,J}[d+1] + w(I) + w(J) - w(I,J). \tag{4}$$

The auxiliary recurrences $L_{I,J}, M_{I,J}, R_{I,J}$ are defined for the three independent regions in the very same way as S_I above with the exception that $L_{I,J}[e]=0, M_{I,J}[d]=0$ and $R_{I,J}[f]=0$. Hence, following essentially the same proof as in Lemma 7 we obtain

▶ **Lemma 8.** *Let $\mathcal{I}$ be a set of intervals and $I,J \in \mathcal{I}$ with $J \in \overrightarrow{\mathcal{P}}(I,\mathcal{I})$, then 1MWOS$(I,J)$ can be computed in $O(\gamma \operatorname{span}(I,J))$ time assuming all 1MWOS(K) and 1MWOS(K,L) values are computed for $K,L \in \mathcal{I}$, $\ell(K)<\operatorname{fit}(I,J)$ and $\operatorname{span}(K,L)<\operatorname{fit}(I,J)$.*

▶ **Lemma 9.** *Let $\mathcal{I}$ be a set of intervals. The 1MWOS values for all $I \in \mathcal{I}$ and all pairs $I,J \in \mathcal{I}$ with $J \in \overrightarrow{\mathcal{P}}(I,\mathcal{I})$ can be computed in $O(\gamma^2\ell)$ time.*

Proof. For an interval $I \in \mathcal{I}$ the value 1MWOS(I) is computed in $O(\gamma\,\ell(I))$ time by Lemma 7. With $\ell = \sum_{I\in\mathcal{I}} \ell(I)$ the claim follows for all $I \in \mathcal{I}$.

By Lemma 8 the value 1MWOS(I,J) can be computed in $O(\gamma \operatorname{span}(I,J))$ time for each overlapping pair I,J with $J \in \overrightarrow{\mathcal{P}}(I,\mathcal{I})$. Since $\operatorname{span}(I,J) \le \ell(I)+\ell(J)$ the time bound of $O(\gamma^2\ell)$ follows by applying Lemma 6. ◀

In the second phase of our algorithm we compute the maximum weight of a 1-overlap set for $\mathcal{I}$ by defining another recurrence $T[x]$ for $x \in \sigma(\mathcal{I})$ and re-using the 1MWOS values. The recurrence for T is defined similarly to the recurrence of S_I above. We set $T[2n]=0$. Let $I=[a,b] \in \mathcal{I}$ be an interval and $b \neq 2n$, then

$$T[b] = T[b+1] \qquad\qquad T[a] = \max\left\{\begin{array}{l}\{T[a+1]\} \cup \\ \{\text{1MWOS}([a,b]) + T[b+1]\} \cup \\ D_T(I,\mathcal{I})\end{array}\right\}, \tag{5}$$

where D_T is defined analogously to D_S by replacing the recurrence S_I with T in eq. (2). The maximum weight of a 1-overlap set for $\mathcal{I}$ is found in $T[1]$.

▶ **Theorem 10.** *A max-weight 1-overlap set for a set of intervals $\mathcal{I}$ can be computed in $O(\gamma^2\ell) \subseteq O(|\mathcal{I}|^4)$ time, where ℓ is the total interval length and γ is the maximum degree of the corresponding overlap graph.*

Proof. The time to compute all 1MWOS values is $O(\gamma^2\ell)$ with Lemma 9. As argued the optimal solution is found by computing $T[1]$. The time to compute T in Recurrence (5) is again dominated by the maximization, which itself is dominated by the evaluation of the D_T sets. The size of these sets is exactly γ times the sum we bounded in Lemma 6. So the total time to compute T is $O(\gamma^2\ell)$. Hence the total running time is $O(\gamma^2\ell)$. From $\gamma \leq |\mathcal{I}|$ and $\ell \leq |\mathcal{I}|^2$ we obtain the coarser bound $O(|\mathcal{I}|^4)$.

It remains to show the correctness of Recurrence (5). Again we can treat it with the same induction used in the proof of Lemma 7. To see this we introduce an interval $[0, 2n+1]$ with weight zero. Now the computation of the maximum weight of a 1-overlap set for $\mathcal{I}$ is the same as computing all 1MWOS values for the instance $\mathcal{I} \cup \{[0, 2n+1]\}$. Using standard backtracking, the same algorithm can be used to compute the max-weight 1-overlap set instead of only its weight. ◀

4.4 An XP-algorithm for max-weight k-overlap sets

In this section we generalize our algorithm to $k \geq 2$. While it is not possible to directly generalize Recurrences (3) and (5) we do use similar concepts. The difficulty for $k > 1$ is that the solution can have arbitrarily large connected parts, e.g., a 2-overlap set can include arbitrarily long paths and cycles. So we can no longer partition an instance along connected components into a constant number of independent sub-instances as we did for the case $k = 1$. Due to space constraints we only sketch the main ideas here and refer the reader to the full version for all omitted proofs.

We first generalize the definition of 1MWOS. Let $\mathcal{I}$ be a set of n intervals as before and $I = [a, b] \in \mathcal{I}$. We define the value $k\text{MWOS}(I)$ as the maximum weight of a k-overlap set on $\mathcal{I}[a, b]$ that includes I in the k-overlap set (if one exists). When computing such a value $k\text{MWOS}(I)$, we consider all subsets $\mathcal{J} \subseteq \mathcal{P}(I, \mathcal{I})$ of cardinality $|\mathcal{J}| \leq k$ of at most k neighbors of I to be included in a k-overlap set, while $\mathcal{P}(I, \mathcal{I}) \setminus \mathcal{J}$ is excluded.

For keeping track of how many intervals are still allowed to overlap each interval we introduce the *capacity* of each interval boundary $i \in \sigma(\mathcal{I}) = \{1, 2, \ldots, 2n\}$. These capacities are stored in a vector $\lambda = (\lambda_1, \ldots, \lambda_{2n})$, where each λ_i is the capacity of the interval boundary $i \in \sigma(\mathcal{I})$. Each λ_i is basically a value in the set $\{0, 1, \ldots, k\}$ that indicates how many additional intervals may still overlap the interval corresponding to i, see Fig. 5. We actually define $k\text{MWOS}_\lambda([a, b])$ as the maximum weight of a k-overlap set in $\mathcal{I}[a, b]$ with pre-defined capacities λ. In the full version of the paper we prove that the number of relevant vectors λ to consider for each interval can be bounded by $O(\gamma^k)$, where γ is the maximum degree of the overlap graph corresponding to $\mathcal{I}$.

For our recursive definition we assume that when computing $k\text{MWOS}_\lambda(I)$ all values $k\text{MWOS}_\lambda(J)$ with $J \in \mathcal{I}$ and $\ell(J) < \ell(I)$ are already computed. The following recurrence computes one $k\text{MWOS}_\lambda(I)$ value given a valid capacity vector λ and an interval $I = [a, b] \in \mathcal{I}$

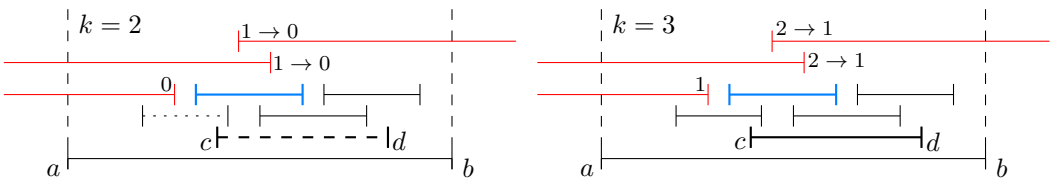


Figure 5 Examples for $k = 2$ and $k = 3$. The red intervals are in a solution set. The arrows indicate how the capacities change if the blue interval is included in a solution. For $k = 2$ we cannot use the interval $[c, d]$ since some capacities are zero, but for $k = 3$ it remains possible.

$$k\text{MWOS}_\lambda([a,b]) = S_{I,\lambda}[a+1] + w([a,b]). \tag{6}$$

This means that we select I for the k-overlap set, add its weight $w(I)$, and recursively solve the subinstance of intervals nested in I subject to the capacities λ. As in the approach for the 1MWOS values the main work is done in recurrence $S_{I,\lambda}[x]$ where $x \in \sigma(\mathcal{I}[a,b])$. In the full version of the paper we prove the correctness of this computation using a similar induction-based proof as Lemma 7 for $k = 1$, but being more careful with the computation of the correct weights. Lemma 11 is a simplified version of this.

▶ **Lemma 11.** *Let $\mathcal{I}$ be a set of intervals, $I \in \mathcal{I}$, λ a valid capacity vector for I, and γ the maximum degree of the corresponding overlap graph. Then kMWOS$_\lambda(I)$ can be computed in $O(\gamma^k \ell(I))$ time once the kMWOS$_\lambda(J)$ values are computed for all $J \in \mathcal{I}$ with $\ell(J) < \ell(I)$.*

Applying Lemma 11 to all $I \in \mathcal{I}$ and all valid capacity vectors λ results in a running time of $O(\gamma^{2k}\ell)$ to compute all $k\text{MWOS}_\lambda(I)$ values, where ℓ is the total interval length.

Now that we know how to compute all values $k\text{MWOS}_\lambda(I)$ for all $I \in \mathcal{I}$ and all relevant capacity vectors λ, we can obtain the optimal solution by introducing a dummy interval $\hat{I}$ with weight $w(\hat{I}) = 0$ that nests the entire set $\mathcal{I}$. We compute the value $k\text{MWOS}_{\hat{\lambda}}(\hat{I})$ for a capacity vector $\hat{\lambda}$ that puts no prior restrictions on the intervals in $\mathcal{I}$. This solution obviously contains the max-weight k-overlap set for $\mathcal{I}$. We summarize:

▶ **Theorem 12.** *A max-weight k-overlap set for a set of intervals $\mathcal{I}$ can be computed in $O(\gamma^{2k}\ell) \subseteq O(|\mathcal{I}|^{2k+2})$ time, where ℓ is the total interval length and γ is the maximum degree of the corresponding overlap graph.*

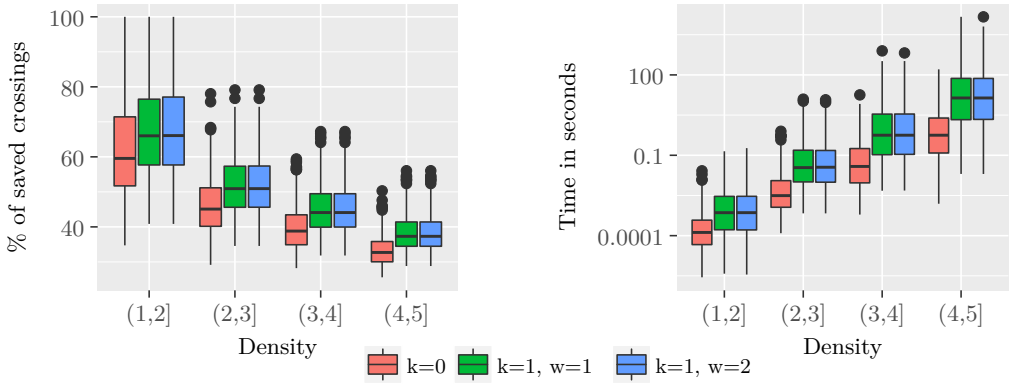
The running time in Theorem 12 implies that both the max-weight k-overlap set problem and the equivalent k-BDMWIS problem for circle (overlap) graphs are in XP.[2] This fact alone can alternatively be derived from a metatheorem of Fomin et al. [6] as follows.[3] The number of minimal separators of circle graphs can be polynomially bounded by $O(n^2)$ as shown by Kloks [14]. Further, since we are interested in a bounded-degree induced subgraph $G[V']$ of a circle graph G, we know from Gaspers et al. [9] that $G[V']$ has treewidth at most four times the maximum degree k. With these two pre-conditions the metatheorem of Fomin et al. [6] yields the existence of an XP-time algorithm for k-BDMWIS on circle graphs. However, the running time obtained from Fomin et al. [6] is $O(|\Pi_G| \cdot n^{t+4} \cdot f(t,\phi))$ where $|\Pi_G|$ is the number of potential cliques in G, t is the treewidth of $G[V']$ with $V' \subseteq V$ being the solution set, and f is a tower function depending only on t and the CMSO (Counting Monadic Second Order Logic) formula ϕ (compare Thomas [22] proving this already for MSO formulas). Let k be the desired degree of a k-BDMWIS instance, then the treewidth of $G[V']$ is at most $4k$. Further by Kloks [14] we know $|\Pi_G| = O(n^2)$. Hence the running-time of the algorithm would be in $O(n^{4k+6} \cdot f(4k,\phi))$, whereas our problem-specific algorithm has running time $O(n^{2k+2})$.

5 Experiments

We implemented the algorithm for 1-BDMWIS from Section 4.3 and the independent set algorithm for 0-BDMWIS in C++. The compiler was g++, version 7.2.0 with set -O3 flag.

[2] The class XP contains problems that can be solved in time $O(n^{f(k)})$, where n is the input size, k is a parameter, and f is a computable function.

[3] We thank an anonymous reviewer of an earlier version for pointing us to this fact.



(a) Density vs the percentage of saved crossings. **(b)** Density vs the computation time.

Figure 6 Plots for the test set with 4881 graphs. w is the weight given to the edges of the circle graph, see Section 2.1. This test set has between 20 and 60 vertices and an average density of 2.6.

Further we used the OGDF library [4] in its most current snapshot version. Experiments were run on a standard desktop computer with an eight core Intel i7-6700 CPU clocked at 3.4 GHz and 16 GB RAM, running Archlinux and kernel version 4.13.12. The implementation is available under `https://www.ac.tuwien.ac.at/two-sided-layouts/`.

We generated two sets of random biconnected graphs using OGDF. The first set has $5,156$ and the second $4,822$ different non-planar graphs. We varied the edge to vertex ratio between 1.0 and 5.0 and the number of vertices between 20 and 60. In addition, we used the Rome graph library (`http://www.graphdrawing.org/data.html`) consisting of $8,504$ non-planar graphs with a density of 0.5 to 2.1 and 10 to 100 vertices. The relative sparsity of the test instances is not a drawback, since it is impractical to use circular layouts for visualizing very dense graphs. For dense graphs one would have to apply some form of bundling strategy to reduce edge clutter. Given such a bundled layout one could consider minimizing bundled crossings [1] and adapt our algorithms to the resulting "bundled" circle graph.

For the random test graphs Fig. 6a displays the percentage of crossings saved by the layouts with exterior edges versus the one-sided circular layout implemented in OGDF. Compared to the approach without exterior crossings we find that allowing up to one crossing per edge in the exterior one can save around 11% more crossings on average for the random instances and 7.5% for the Rome graphs. Setting the edge weight in 1-BDMWIS to one (i.e., counting exterior crossings in the optimization) or two (i.e., not counting exterior crossings in the optimization), has (almost) no noticeable effect.

Figure 6b depicts the times needed to compute the layouts for the respective densities. We observe the expected behaviour. The case of $k = 0$ with $O(|\mathcal{E}|^2)$ time is a lot faster as the graphs get more dense. Still for our sparse test instances our algorithm for 1-BDMWIS with $O(|\mathcal{E}|^4)$ time runs sufficiently fast to be used on graphs with up to 60 vertices (82 seconds on average). For additional plots we refer to the full version of the paper.

Our tests show a clear improvement in crossing reduction when going from $k = 0$ to $k = 1$. Of course this comes with a non-negligible runtime increase. For the practically interesting sparse instances, though, 1-BDMWIS can be solved fast enough to be useful in practice.

6 Open questions

The overall hardness of the k-BDMWIS problem on circle graphs, parametrized by just the desired degree k remains open. While we could show NP-hardness, we do not know whether an FPT-algorithm exists or whether the problem is W[1]-hard. In terms of the motivating graph layout problem crossing minimization is known as a major factor for readability. Yet, practical two-sided layout algorithms must also apply suitable vertex-ordering heuristics and they should further take into account the length and actual routing of exterior edges. Edge bundling approaches for both interior and exterior edges promise to further reduce visual clutter, but then bundling and bundled crossing minimization should be considered simultaneously. It would also be interesting to generalize the problem from circular layouts to other layout types, where many but not necessarily all vertices can be fixed on a boundary curve.

References

1 Md Jawaherul Alam, Martin Fink, and Sergey Pupyrev. The bundled crossing number. In Y. Hu and M. Nöllenburg, editors, *Graph Drawing and Network Visualization (GD'16)*, volume 9801 of *LNCS*, pages 399–412. Springer, 2016.

2 Michael Baur and Ulrik Brandes. Crossing reduction in circular layouts. In *Graph-Theoretic Concepts in Computer Science (WG'04)*, volume 3353 of *LNCS*, pages 332–343. Springer, 2004.

3 Nadja Betzler, Robert Bredereck, Rolf Niedermeier, and Johannes Uhlmann. On bounded-degree vertex deletion parameterized by treewidth. *Discrete Applied Mathematics*, 160(1–2):53–60, 2012.

4 Markus Chimani, Carsten Gutwenger, Michael Jünger, Gunnar W Klau, Karsten Klein, and Petra Mutzel. The open graph drawing framework (OGDF). In R. Tamassia, editor, *Handbook of Graph Drawing and Visualization*, chapter 17, pages 543–569. CRC Press, 2013.

5 Anders Dessmark, Klaus Jansen, and Andrzej Lingas. The maximum k-dependent and f-dependent set problem. In K. W. Ng, P. Raghavan, N. V. Balasubramanian, and F. Y. L. Chin, editors, *Algorithms and Computation (ISAAC'93)*, volume 762 of *LNCS*, pages 88–97. Springer, 1993.

6 Fedor V Fomin, Ioan Todinca, and Yngve Villanger. Large induced subgraphs via triangulations and CMSO. *SIAM J. Computing*, 44(1):54–87, 2015.

7 Robert Ganian, Fabian Klute, and Sebastian Ordyniak. On Structural Parameterizations of the Bounded-Degree Vertex Deletion Problem. In *Theoretical Aspects of Computer Science (STACS'18)*, volume 96 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 33:1–33:14, 2018.

8 Emden R Gansner and Yehuda Koren. Improved circular layouts. In *Graph Drawing (GD'06)*, volume 4372 of *LNCS*, pages 386–398. Springer, 2007.

9 Serge Gaspers, Dieter Kratsch, Mathieu Liedloff, and Ioan Todinca. Exponential time algorithms for the minimum dominating set problem on some graph classes. *ACM Trans. Algorithms*, 6(1):9:1–9:21, 2009.

10 Fanika Gavril. Algorithms for a maximum clique and a maximum independent set of a circle graph. *Networks*, 3(3):261–273, 1973.

11 Michael Jünger and Petra Mutzel, editors. *Graph Drawing Software*. Mathematics and Visualization. Springer, 2004.

12 J Mark Keil. The complexity of domination problems in circle graphs. *Discrete Applied Mathematics*, 42(1):51–63, 1993.

13 Jonathan Klawitter, Tamara Mchedlidze, and Martin Nöllenburg. Experimental evaluation of book drawing algorithms. In F. Frati and K.-L. Ma, editors, *Graph Drawing and Network Visualization (GD'17)*, volume 10692 of *LNCS*, pages 224–238. Springer, 2018. URL: 1708.09221.

14 Ton Kloks. Treewidth of circle graphs. *International J. Foundations of Computer Science*, 7(02):111–120, 1996.

15 Stephen G. Kobourov, Giuseppe Liotta, and Fabrizio Montecchiani. An annotated bibliography on 1-planarity. *CoRR*, abs/1703.02261, 2017. arXiv:1703.02261.

16 Martin I. Krzywinski, Jacqueline E. Schein, Inanc Birol, Joseph Connors, Randy Gascoyne, Doug Horsman, Steven J. Jones, and Marco A. Marra. Circos: An information aesthetic for comparative genomics. *Genome Research*, 19:1639–1645, 2009.

17 Giuseppe Liotta. Graph drawing beyond planarity: some results and open problems. In *Italian Conference on Theoretical Computer Science (ICTCS'14)*, pages 3–8, 2014.

18 Sumio Masuda, Toshinobu Kashiwabara, Kazuo Nakajima, and Toshio Fujisawa. On the NP-completeness of a computer network layout problem. In *Circuits and Systems (ISCAS'87)*, pages 292–295. IEEE, 1987.

19 Sumio Masuda, Kazuo Nakajima, Toshinobu Kashiwabara, and Toshio Fujisawa. Crossing minimization in linear embeddings of graphs. *IEEE Trans. Computers*, 39(1):124–127, 1990.

20 Farhad Shahrokhi, László A. Székely, Ondrej Sýkora, and Imrich Vrt'o. The book crossing number of a graph. *J. Graph Theory*, 21(4):413–424, 1996.

21 Janet M. Six and Ioannis G. Tollis. Circular drawing algorithms. In R. Tamassia, editor, *Handbook of Graph Drawing and Visualization*, chapter 9, pages 285–315. CRC Press, 2013.

22 Wolfgang Thomas. Languages, automata, and logic. *Handbook of formal languages*, 3:389–455, 1996.

23 Gabriel Valiente. A new simple algorithm for the maximum-weight independent set problem on circle graphs. In *Algorithms and Computation (ISAAC'03)*, volume 2906 of *LNCS*, pages 129–137. Springer, 2003.

24 Mihalis Yannakakis. Node- and edge-deletion NP-complete problems. In *Theory of Computing (STOC'78)*, pages 253–264. ACM, 1978.

Discrete Stratified Morse Theory: A User's Guide

Kevin Knudson
Department of Mathematics, University of Florida
Gainesville, FL, USA
kknudson@ufl.edu
https://orcid.org/0000-0001-6768-2542

Bei Wang[1]
School of Computing, Scientific Computing and Imaging Institute, University of Utah
Salt Lake City, UT, USA
beiwang@sci.utah.edu
https://orcid.org/0000-0002-9240-0700

Abstract

Inspired by the works of Forman on discrete Morse theory, which is a combinatorial adaptation to cell complexes of classical Morse theory on manifolds, we introduce a discrete analogue of the stratified Morse theory of Goresky and MacPherson. We describe the basics of this theory and prove fundamental theorems relating the topology of a general simplicial complex with the critical simplices of a discrete stratified Morse function on the complex. We also provide an algorithm that constructs a discrete stratified Morse function out of an arbitrary function defined on a finite simplicial complex; this is different from simply constructing a discrete Morse function on such a complex. We borrow Forman's idea of a "user's guide," where we give simple examples to convey the utility of our theory.

2012 ACM Subject Classification Theory of computation → Randomness, geometry and discrete structures, Mathematics of computing → Combinatorics

Keywords and phrases Discrete Morse theory, stratified Morse theory, topological data analysis

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.54

Related Version A full version is available at [20], `https://arxiv.org/abs/1801.03183`.

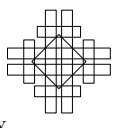
Acknowledgements BW would like to thank Paul Bendich for a discussion on this topic in 2012.

1 Introduction

It is difficult to overstate the utility of classical Morse theory in the study of manifolds. A Morse function $f : \mathbb{M} \to \mathbb{R}$ determines an enormous amount of information about the manifold $\mathbb{M}$: a handlebody decomposition, a realization of $\mathbb{M}$ as a CW-complex whose cells are determined by the critical points of f, a chain complex for computing the integral homology of $\mathbb{M}$, and much more.

With this as motivation, Forman developed discrete Morse theory on general cell complexes [11]. This is a combinatorial theory in which function values are assigned not to points in a space but rather to entire cells. Such functions are not arbitrary; the defining conditions require that function values generically increase with the dimensions of the cells in the complex. Given a cell complex with set of cells K, a discrete Morse function $f : K \to \mathbb{R}$ yields information about the cell complex similar to what happens in the smooth case.

[1] NSF IIS-1513616 and NSF ABI-1661375



While the category of manifolds is rather expansive, it is not sufficient to describe all situations of interest. Sometimes one is forced to deal with singularities, most notably in the study of algebraic varieties. One approach to this is to expand the class of functions one allows, and this led to the development of stratified Morse theory by Goresky and MacPherson [15]. The main objects of study in this theory are *Whitney stratified spaces*, which decompose into pieces that are smooth manifolds. Such spaces are triangulable.

The goal of this paper is to generalize stratified Morse theory to finite simplicial complexes, much as Forman did in the classical smooth case. Given that stratified spaces admit simplicial structures, and any simplicial complex admits interesting discrete Morse functions, this could be the end of the story. However, we present examples in this paper illustrating that the class of discrete stratified Morse functions defined here is much larger than that of discrete Morse functions. Moreover, there exist discrete stratified Morse functions that are nontrivial and interesting from a data analysis point of view. Our motivations are three-fold.

1. **Generating discrete stratified Morse functions from point cloud data.** Consider the following scenario. Suppose K is a simplicial complex and that f is a function defined on the 0-skeleton of K. Such functions arise naturally in data analysis where one has a sample of function values on a space. Algorithms exist to build discrete Morse functions on K extending f (see, for example, [18]). Unfortunately, these are often of potentially high computational complexity and might not behave as well as we would like. In our framework, we may take this input and generate a discrete stratified Morse function which will not be a global discrete Morse function in general, but which will allow us to obtain interesting information about the underlying complex.
2. **Filtration-preserving reductions of complexes in persistent homology and parallel computation.** As discrete Morse theory is useful for providing a filtration-preserving reduction of complexes in the computation of both persistent homology [6, 21, 25] and multi-parameter persistent homology [1], we believe that discrete stratified Morse theory could help to push the computational boundary even further. First, given any real-valued function $f : K \to \mathbb{R}$, defined on a simplicial complex, our algorithm generates a stratification of K such that the restriction of f to each stratum is a discrete Morse function. Applying *Morse pairing* to each stratum reduces K to a smaller complex of the same homotopy type. Second, if such a reduction can be performed in a filtration-preserving way with respect to each stratum, it would lead to a faster computation of persistent homology in the setting where the function is not required to be Morse. Finally, since discrete Morse theory can be applied independently to each stratum of K, we can design a parallel algorithm that computes persistent homology pairings by strata and uses the stratification (i.e. relations among strata) to combine the results.
3. **Applications in imaging and visualization.** Discrete Morse theory can be used to construct discrete Morse complexes in imaging (e.g. [5, 25]), as well as Morse-Smale complexes [7, 8] in visualization (e.g. [16, 17]). In addition, it plays an essential role in the visualization of scalar fields and vector fields (e.g. [23, 24]). Since discrete stratified Morse theory leads naturally to stratification-induced domain partitioning where discrete Morse theory becomes applicable, we envision our theory to have wide applicability for the analysis and visualization of large complex data.

Contributions. Throughout the paper, we hope to convey via simple examples the usability of our theory. It is important to note that our discrete stratified Morse theory is *not* a simple reinterpretation of discrete Morse theory; it considers a larger class of functions defined on any finite simplicial complex and has potentially many implications for data analysis. Our

contributions are:

1. We describe the basics of a discrete stratified Morse theory and prove fundamental theorems that relate the topology of a finite simplicial complex with the critical simplices of a discrete stratified Morse function defined on the complex.
2. We provide an algorithm that constructs a discrete stratified Morse function on any finite simplicial complex equipped with a real-valued function.

A simple example. We begin with an example from [13], where we demonstrate how a discrete stratified Morse function can be constructed from a function that is not a discrete Morse function. As illustrated in Figure 1, the function on the left is a discrete Morse function where the green arrows can be viewed as its discrete gradient vector field; function f in the middle is not a discrete Morse function, as the vertex $f^{-1}(5)$ and the edge $f^{-1}(0)$ both violate the defining conditions of a discrete Morse function. However, we can equip f with a stratification s by treating such violators as their own independent strata, therefore converting it into a discrete stratified Morse function.

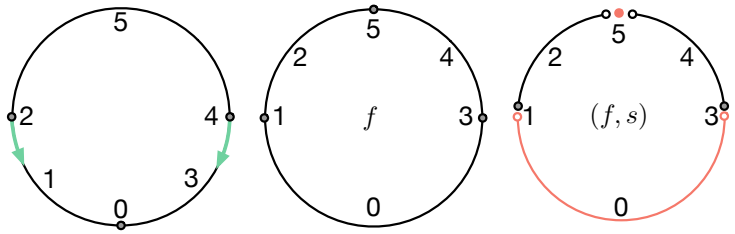


Figure 1 The function on the left is a discrete Morse function. The function f in the middle is not a discrete Morse function; however, it can be converted into a discrete stratified Morse function when it is equipped with an appropriate stratification s.

2 Preliminaries on discrete Morse theory

We review the most relevant definitions and results on discrete Morse theory and refer the reader to the full version [20] for a review of classical Morse theory. Discrete Morse theory is a combinatorial version of Morse theory [11, 13]. It can be defined for any CW complex but in this paper we will restrict our attention to finite simplicial complexes.

Discrete Morse functions. Let K be any finite simplicial complex, where K need not be a triangulated manifold nor have any other special property [12]. When we write K we mean the set of simplices of K; by $|K|$ we mean the underlying topological space. Let $\alpha^{(p)} \in K$ denote a simplex of dimension p. Let $\alpha < \beta$ denote that simplex α is a face of simplex β. If $f : K \to \mathbb{R}$ is a function define $U(\alpha) = \{\beta^{(p+1)} > \alpha \mid f(\beta) \leq f(\alpha)\}$ and $L(\alpha) = \{\gamma^{(p-1)} < \alpha \mid f(\gamma) \geq f(\alpha)\}$. In other words, $U(\alpha)$ contains the immediate cofaces of α with lower (or equal) function values, while $L(\alpha)$ contains the immediate faces of α with higher (or equal) function values. Let $|U(\alpha)|$ and $|L(\alpha)|$ be their sizes.

▶ **Definition 1.** A function $f : K \to \mathbb{R}$ is a *discrete Morse function* if for every $\alpha^{(p)} \in K$, (i) $|U(\alpha)| \leq 1$ and (ii) $|L(\alpha)| \leq 1$.

Forman showed that conditions (i) and (ii) are exclusive – if one of the sets $U(\alpha)$ or $L(\alpha)$ is nonempty then the other one must be empty ([11], Lemma 2.5). Therefore each simplex

$\alpha \in K$ can be paired with at most one exception simplex: either a face γ with larger function value, or a coface β with smaller function value. Formally, this means that if K is a simplicial complex with a discrete Morse function f, then for any simplex α, either (i) $|U(\alpha)| = 0$ or (ii) $|L(\alpha)| = 0$ ([13], Lemma 2.4).

▶ **Definition 2.** A simplex $\alpha^{(p)}$ is *critical* if (i) $|U(\alpha)| = 0$ and (ii) $|L(\alpha)| = 0$. A *critical value* of f is its value at a critical simplex.

▶ **Definition 3.** A simplex $\alpha^{(p)}$ is *noncritical* if either of the following conditions holds: (i) $|U(\alpha)| = 1$; (ii) $|L(\alpha)| = 1$; as noted above these conditions can not both be true ([11], Lemma 2.5).

Given $c \in \mathbb{R}$, we have the *level subcomplex* $K_c = \cup_{f(\alpha)\leq c} \cup_{\beta\leq\alpha} \beta$. That is, K_c contains all simplices α of K such that $f(\alpha) \leq c$ along with all of their faces.

Results. We have the following two combinatorial versions of the main results of classical Morse theory (see the full version).

▶ **Theorem 4** (DMT Part A, [12])**.** *Suppose the interval $(a, b]$ contains no critical value of f. Then K_b is homotopy equivalent to K_a. In fact, K_b simplicially collapses onto K_a.*

A key component in the proof of Theorem 4 is the following fact [11]: for a simplicial complex equipped with an arbitrary discrete Morse function, when passing from one level subcomplex to the next, the noncritical simplices are added in pairs, each of which consists of a simplex and a free face.

The next theorem explains how the topology of the sublevel complexes changes as one passes a critical value of a discrete Morse function. In what follows, $\dot{e}^{(p)}$ denotes the boundary of a p-simplex $e^{(p)}$. Adjunction spaces, such as the space appearing in this result, are defined in Section 3.1 below.

▶ **Theorem 5** (DMT Part B, [12])**.** *Suppose $\sigma^{(p)}$ is a critical simplex with $f(\sigma) \in (a, b]$, and there are no other critical simplices with values in $(a, b]$. Then K_b is homotopy equivalent to attaching a p-cell $e^{(p)}$ along its entire boundary in K_a; that is, $K_b = K_a \cup_{\dot{e}^{(p)}} e^{(p)}$.*

The associated gradient vector field. Given a discrete Morse function $f : K \to \mathbb{R}$ we may associate a discrete gradient vector field as follows. Since any noncritical simplex $\alpha^{(p)}$ has at most one of the sets $U(\alpha)$ and $L(\alpha)$ nonempty, there is a unique face $\nu^{(p-1)} < \alpha$ with $f(\nu) \geq f(\alpha)$ or a unique coface $\beta^{(p+1)} > \alpha$ with $f(\beta) \leq f(\alpha)$. Denote by V the collection of all such pairs $\{\sigma < \tau\}$. Then every simplex in K is in at most one pair in V and the simplices not in any pair are precisely the critical cells of the function f. We call V the *gradient vector field associated to f*. We visualize V by drawing an arrow from α to β for every pair $\{\alpha < \beta\} \in V$. Theorems 4 and 5 may then be visualized in terms of V by collapsing the pairs in V using the arrows. Thus a discrete gradient (or equivalently a discrete Morse function) provides a collapsing order for the complex K, simplifying it to a complex L with potentially fewer cells but having the same homotopy type.

The collection V has the following property. By a *V-path*, we mean a sequence

$$\alpha_0^{(p)} < \beta_0^{(p+1)} > \alpha_1^{(p)} < \beta_1^{(p+1)} > \cdots < \beta_r^{(p+1)} > \alpha_{r+1}^{(p)}$$

where each $\{\alpha_i < \beta_i\}$ is a pair in V. Such a path is *nontrivial* if $r > 0$ and *closed* if $\alpha_{r+1} = \alpha_0$. Forman proved the following result.

▶ **Theorem 6** ([11]). *If V is a gradient vector field associated to a discrete Morse function f on K, then V has no nontrivial closed V-paths.*

In fact, if one defines a discrete vector field W to be a collection of pairs of simplices of K such that each simplex is in at most one pair in W, then one can show that if W has no nontrivial closed W-paths there is a discrete Morse function f on K whose associated gradient is precisely W.

3 A discrete stratified Morse theory

Our goal is to describe a combinatorial version of stratified Morse theory. To do so, we need to: (a) define a discrete stratified Morse function; and (b) prove the combinatorial versions of the relevant fundamental results. Our results are very general as they apply to any finite simplicial complex K equipped with a real-valued function $f : K \to \mathbb{R}$. Our work is motivated by relevant concepts from (classical) stratified Morse theory [15], whose details are found in the full version.

3.1 Background

Open simplices. To state our main results, we need to consider open simplices (as opposed to the closed simplices of Section 2). Let $\{a_0, a_1, \cdots, a_k\}$ be a geometrically independent set in $\mathbb{R}^N$, a *closed k-simplex* $[\sigma]$ is the set of all points x of $\mathbb{R}^N$ such that $x = \sum_{i=0}^{k} t_i a_i$, where $\sum_{i=0}^{k} t_i = 1$ and $t_i \geq 0$ for all i [22]. An *open simplex* (σ) is the interior of the closed simplex $[\sigma]$.

A *simplicial complex* K is a finite set of open simplices such that: (a) If $(\sigma) \in K$ then all open faces of $[\sigma]$ are in K; (b) If $(\sigma_1), (\sigma_2) \in K$ and $(\sigma_1) \cap (\sigma_2) \neq \emptyset$, then $(\sigma_1) = (\sigma_2)$. For the remainder of this paper, we always work with a finite open simplicial complex K.

Unless otherwise specified, we work with open simplices σ and define the boundary $\dot{\sigma}$ to be the boundary of its closure. We will often need to talk about a "half-open" or "half-closed" simplex, consisting of the open simplex σ along with some of the open faces in its boundary $\dot{\sigma}$. We denote such objects ambiguously as $[\sigma)$ or $(\sigma]$, specifying particular pieces of the boundary as necessary.

Stratified simplicial complexes. A simplicial complex K equipped with a stratification is referred to as a *stratified simplicial complex.* [2] A *stratification* of a simplicial complex K is a finite filtration

$$\emptyset = K^0 \subset K^1 \subset \cdots \subset K^m = K,$$

such that for each i, $K^i - K^{i-1}$ is a locally closed subset of K. [3] We say a subset $L \subset K$ is locally closed if it is the intersection of an open and a closed set in K. We will refer to a connected component of the space $K^i - K^{i-1}$ as a *stratum*; and the collection of all strata is denoted by $\mathcal{S} = \{S_j\}$. We may consider a stratification as an assignment from K to the set $\mathcal{S}$, denoted $s : K \to \mathcal{S}$.

In our setting, each S_j is the union of finitely many open simplices (that may not form a subcomplex of K); and each open simplex σ in K is assigned to a particular stratum $s(\sigma)$ via the mapping s.

[2] Our notion of a stratified simplicial complex can be considered as a relaxed version of the notion in [2].
[3] Technically we should speak of the geometric realization $|K^i - K^{i-1}|$ being a locally closed subspace of $|K|$; we often confuse these notations as it should be clear from context.

Adjunction spaces. Let X and Y be topological spaces with $A \subseteq X$. Let $f : A \to Y$ be a continuous map called the *attaching* map. The *adjunction space* $X \cup_f Y$ is obtained by taking the disjoint union of X and Y by identifying x with $f(x)$ for all x in A. That is, Y is *glued* onto X via a quotient map, $X \cup_f Y = (X \amalg Y)/\{f(A) \sim A\}$. We sometimes abuse the notion as $X \cup_A Y$, when f is clear from the context (e.g. an inclusion).

Gluing theorem for homotopy equivalences. In homotopy theory, a continuous mapping $i : A \to X$ is a *cofibration* if there is a retraction from $X \times I$ to $(A \times I) \cup (X \times \{0\})$. In particular, this holds if X is a cell complex and A is a subcomplex of X; it follows that the inclusion $i : A \to X$ a closed cofibration.

▶ **Theorem 7** (Gluing theorem for adjunction spaces ([4], Theorem 7.5.7))**.** *Suppose we have the following commutative diagram of topological spaces and continuous maps:*

$$\begin{array}{ccccc} Y & \xleftarrow{f} & A & \xhookrightarrow{i} & X \\ \downarrow{\scriptstyle \varphi_Y} & & \downarrow{\scriptstyle \varphi_A} & & \downarrow{\scriptstyle \varphi_X} \\ Y' & \xleftarrow{f'} & A' & \xhookrightarrow{i'} & X' \end{array}$$

where φ_A, φ_X and φ_Y are homotopy equivalences and inclusions i and i' are closed cofibrations, then the map $\phi : X \cup_f Y \to X' \cup_f Y'$ induced by ϕ_A, ϕ_X ad ϕ_Y is a homotopy equivalence.

In our setting, since we are not in general dealing with closed subcomplexes of simplicial complexes, this theorem does not apply directly. However, the condition that the maps i, i' be closed cofibrations is not necessary (see [26], 5.3.2, 5.3.3), and in our setting it will be the case that our various pairs (X, A) will satisfy the property that $X \times \{0\} \cup A \times [0, 1]$ is a retract of $X \times [0, 1]$.

Stratum-preserving homotopies. If X and Y are two filtered spaces, we call a map $f : X \to Y$ *stratum-preserving* if the image of each component of a stratum of X lies in a stratum of Y [14]. A map $f : X \to Y$ is a *stratum-preserving homotopy equivalence* if there exists a stratum-preserving map $g : Y \to X$ such that $g \circ f$ and $f \circ g$ are homotopic to the identity [14].

3.2 A primer

Discrete stratified Morse function. Let K be a simplicial complex equipped with a stratification s and a discrete stratified Morse function $f : K \to \mathbb{R}$. We define

$$U_s(\alpha) = \{\beta^{(p+1)} > \alpha \mid s(\beta) = s(\alpha) \text{ and } f(\beta) \leq f(\alpha)\},$$
$$L_s(\alpha) = \{\gamma^{(p-1)} < \alpha \mid s(\gamma) = s(\alpha) \text{ and } f(\gamma) \geq f(\alpha)\}.$$

▶ **Definition 8.** Given a simplicial complex K equipped with a stratification $s : K \to \mathcal{S}$, a function $f : K \to \mathbb{R}$ (equipped with s) is a *discrete stratified Morse function* if for every $\alpha^{(p)} \in K$, (i) $|U_s(\alpha)| \leq 1$ and (ii) $|L_s(\alpha)| \leq 1$.

In other words, a discrete stratified Morse function is a pair (f, s) where $f : K \to \mathbb{R}$ is a discrete Morse function when restricted to each stratum $S_j \in \mathcal{S}$. We omit the symbol s whenever it is clear from the context.

▶ **Definition 9.** A simplex $\alpha^{(p)}$ is *critical* if (i) $|U_s(\alpha)| = 0$ and (ii) $|L_s(\alpha)| = 0$. A *critical value* of f is its value at a critical simplex.

▶ **Definition 10.** A simplex $\alpha^{(p)}$ is *noncritical* if exactly one of the following two conditions holds: (i) $|U_s(\alpha)| = 1$ and $|L_s(\alpha)| = 0$; or (ii) $|L_s(\alpha)| = 1$ and $|U_s(\alpha)| = 0$.

The two conditions in Definition 10 mean that, within the same stratum as $s(\alpha)$: (i) $\exists \beta^{(p+1)} > \alpha$ with $f(\beta) \leq f(\alpha)$ or (ii) $\exists \gamma^{(p-1)} < \alpha$ with $f(\gamma) \geq f(\alpha)$; conditions (i) and (ii) cannot both be true.

Note that a classical discrete Morse function $f : K \to \mathbb{R}$ is a discrete stratified Morse function with the trivial stratification $\mathcal{S} = \{K\}$. We will present several examples in Section 4 illustrating that the class of discrete stratified Morse functions is much larger.

Violators. The following definition is central to our algorithm in constructing a discrete stratified Morse function from any real-valued function defined on a simplicial complex.

▶ **Definition 11.** Given a simplicial complex K equipped with a real-valued function, $f : K \to \mathbb{R}$. A simplex $\alpha^{(p)}$ is a *violator* of the conditions associated with a discrete Morse function if one of these conditions hold: (i) $|U(\alpha)| \geq 2$; (ii) $|L(\alpha)| \geq 2$; (iii) $|U(\alpha)| = 1$ and $|L(\alpha)| = 1$. These are referred to as type I, II and III violators; the sets containing such violators are not necessarily mutually exclusive.

3.3 Main results

To describe our main results, we work with the *sublevel set* of an open simplicial complex K, where $K_c = \cup_{f(\alpha) \leq c} \alpha$, for any $c \in \mathbb{R}$. That is, K_c contains all open simplices α of K such that $f(\alpha) \leq c$. Note that K_c is not necessarily a subcomplex of K. Suppose that K is a simplicial complex equipped with a stratification s and a discrete stratified Morse function $f : K \to \mathbb{R}$. We now state our two main results which will be proved in Section 5.

▶ **Theorem 12** (DSMT Part A)**.** *Suppose the interval $(a, b]$ contains no critical value of f. Then K_b is stratum-preserving homotopy equivalent to K_a.*

▶ **Theorem 13** (DSMT Part B)**.** *Suppose $\sigma^{(p)}$ is a critical simplex with $f(\sigma) \in (a, b]$, and there are no other critical simplices with values in $(a, b]$. Then K_b is homotopy equivalent to attaching a p-cell $e^{(p)}$ along its boundary in K_a; that is, $K_b = K_a \cup_{\dot{e^{(p)}}|_{K_a}} e^{(p)}$.*

Remarks. K_c as defined above falls under a nonclassical notion of a "simplicial complex" as defined in [19]: K is a "simplicial complex" if it is the union of finitely many open simplices σ_1, σ_2, ...σ_t in some $\mathbb{R}^N$ such that the intersection of the closure of any two simplices σ_i and σ_j is either a common face of them or empty. Thus the closure $[K] = \{[\sigma_i]\}_{i=1}^t$ of K is a classical finite simplicial complex; and K is obtained from $[K]$ by omitting some open faces.

3.4 Algorithm

We give an algorithm to construct a discrete stratified Morse function from any real-valued function on a simplicial complex.

Given a simplicial complex K equipped with a real-valued function, $f : K \to \mathbb{R}$, define a collection of strata $\mathcal{S}$ as follows. Each violator $\sigma^{(p)}$ is an element of the collection $\mathcal{S}$. Let $\mathcal{V}$ denote the set of violators and denote by S_j the connected components of $K \setminus \mathcal{V}$. Then we set $\mathcal{S} = \mathcal{V} \cup \{S_j\}$. Denote by $s : K \to \mathcal{S}$ the assignment of the simplices of K to their corresponding strata.

We realize this as a stratification of K by taking $K^1 = \bigcup_j S_j$ and then adjoining the elements of $\mathcal{V}$ one simplex at a time by increasing function values (we may assume that f is injective). This filtration is unimportant for our purposes; rather, we shall focus on the strata themselves. We have the following theorem whose proof is delayed to Section 5.

▶ **Theorem 14.** *The function f equipped with the stratification s produced by the algorithm above is a discrete stratified Morse function.*

The algorithm described above is rather lazy. An alternative approach would be to remove violators one at a time by increasing dimension, and after each removal, check to see if what remains is a discrete Morse function globally. This requires more computation at each stage, but note that the extra work is entirely local–one need only check simplices adjacent to the removed violator. Example 1 below illustrates how this more aggressive approach can lead to further simplification of the complex.

4 Discrete stratified Morse theory by example

We apply the algorithm described in 3.4 to a collection of examples to demonstrate the utility of our theory. For each example, given an $f : K \to \mathbb{R}$ that is not necessarily a discrete Morse function, we equip f with a particular stratification s, thereby converting it to a discrete stratified Morse function (f, s). These examples help to illustrate that the class of discrete stratified Morse functions is much larger than that of discrete Morse functions.

Example 1: upside-down pentagon. As illustrated in Figure 2 (left), $f : K \to \mathbb{R}$ defined on the boundary of an upside-down pentagon is not a discrete Morse function, as it contains a set of violators: $\mathcal{V} = \{f^{-1}(10), f^{-1}(1), f^{-1}(2)\}$, since $|U(f^{-1}(10))| = 2$ and $|L(f^{-1}(1))| = |L(f^{-1}(2))| = 2$, respectively.

We construct a stratification s by considering elements in $\mathcal{V}$ and connected components in $K \setminus \mathcal{V}$ as their own strata, as shown in Figure 2 (top middle). The resulting discrete stratified Morse function (f, s) is a discrete Morse function when restricted to each stratum.

Recall that a simplex is critical for (f, s) if it is neither the source nor the target of a discrete gradient vector. The critical values of (f, s) are therefore $1, 2, 3, 4, 9$ and 10. The vertex $f^{-1}(3)$ is noncritical for f since $|U(f^{-1}(3))| = 1$ and $|L(f^{-1}(3))| = 0$; however it is critical for (f, s) since $|U_s(f^{-1}(3))| = L_s(f^{-1}(3))| = 0$.

One of the primary uses of classical discrete Morse theory is *simplification*. In this example, we can collapse a portion of each stratum following the discrete gradient field (illustrated by green arrows, see Section 2). Removing the Morse pairs $(f^{-1}(7), f^{-1}(5))$ and

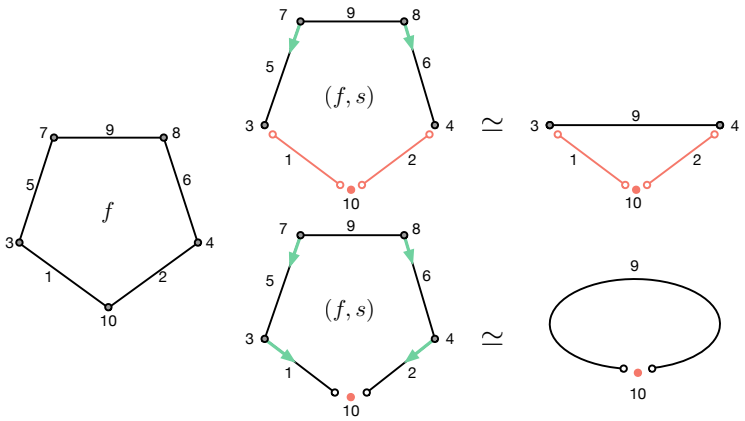


Figure 2 Example 1: upside-down pentagon. Left: f is not a discrete Morse function. Top middle: (f, s) is a discrete stratified Morse function where violators are in red. Top right: the simplified simplicial complex following the discrete gradient vector field (green arrows). Bottom middle and bottom right: the results following a more aggressive algorithm in Section 3.

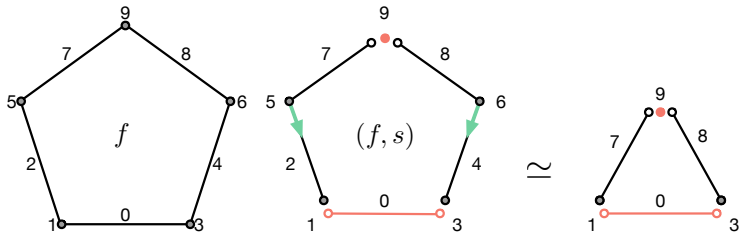


Figure 3 Example 2: pentagon. Middle: there are four strata pieces associated with the discrete stratified Morse function (f, s).

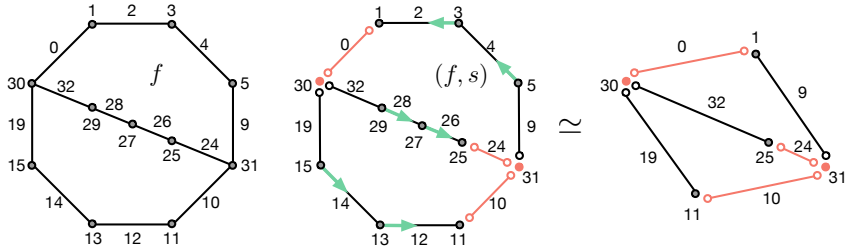


Figure 4 Example 3: split octagon. f is defined on the triangulation of a stratified space.

$(f^{-1}(8), f^{-1}(6))$ simplifies the original complex as much as possible without changing its homotopy type, see Figure 2 (top right).

Note that if we follow the more aggressive algorithm described at the end of Section 3 above, we would first remove the violator $f^{-1}(10)$ and check to see if what remains is a discrete Morse function. In this case, we see that this is indeed the case: we have the additional Morse pairs $(f^{-1}(3), f^{-1}(1))$ and $(f^{-1}(4), f^{-1}(2))$. The resulting simplification yields a complex with one vertex and one edge, see Figure 2 (bottom right).

Example 2: pentagon. For our second pentagon example, f can be made into a discrete stratified Morse function (f, s) by making $f^{-1}(0)$ (a type II violator) and $f^{-1}(9)$ (a type I violator) their own strata (Figure 3). The critical values of (f, s) are $0, 1, 3, 7, 8$ and 9. The simplicial complex can be reduced to one with fewer cells by canceling the Morse pairs, as shown in Figure 3 (right).

Example 3: split octagon. The split octagon example (Figure 4) begins with a function f defined on a triangulation of a stratified space that consists of two 0-dimensional and three 1-dimensional strata. The violators are $f^{-1}(0)$, $f^{-1}(10)$, $f^{-1}(24)$, $f^{-1}(30)$ and $f^{-1}(31)$. The result of canceling Morse pairs yields the simpler complex shown on the right.

Example 4: tetrahedron. In Figure 5, the values of the function f defined on the simplices of a tetrahedron are specified for each dimension. For each simplex $\alpha \in K$, we list the elements of its corresponding $U(\alpha)$ and $L(\alpha)$ in Table 1. We also classify each simplex in terms of its criticality in the setting of classical discrete Morse theory. According to Table 1, violators with function values of $10, 14$ (type I), 6 (type II), $7, 8, 11, 12$ (type III) form their individual strata in (f, s). Given such a stratification s, every simplex is critical except for $f^{-1}(2)$ and $f^{-1}(3)$. Observing that the space is homeomorphic to S^2 and collapsing the single Morse pair $(f^{-1}(2), f^{-1}(3))$ yields a space of the same homotopy type.

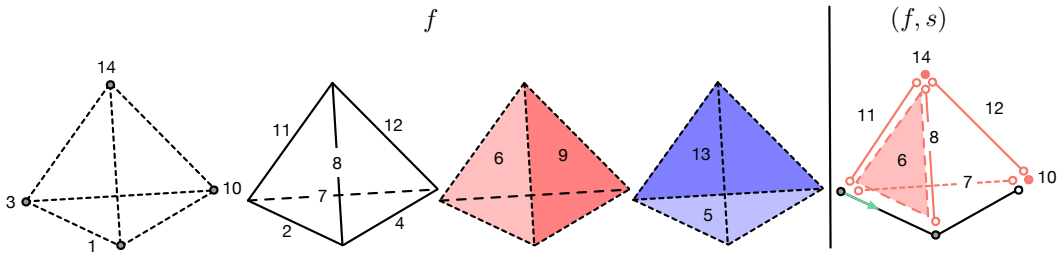


Figure 5 Example 4: tetrahedron. Left: f is defined on the simplices of increasing dimensions. Right: violators are highlighted in red; not all simplicies are shown for (f, s).

Table 1 Example 4: tetrahedron. For simplicity, a simplex α is represented by its function value $f(\alpha)$ (as f is 1-to-1). In terms of criticality for each simplex: C means critical; R means regular; I, II and III correspond to type I, II and III violators.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$U(\alpha)$	$\emptyset$	$\emptyset$	$\{2\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\{15\}$	$\{6\}$	$\emptyset$	$\{4,7\}$	$\{6\}$	$\{9\}$	$\emptyset$	$\{8,11,12\}$
$L(\alpha)$	$\emptyset$	$\{3\}$	$\emptyset$	$\{10\}$	$\{7\}$	$\{8,11\}$	$\{10\}$	$\{14\}$	$\{12\}$	$\emptyset$	$\{14\}$	$\{14\}$	$\emptyset$	$\emptyset$
Type	C	R	R	R	R	II	III	III	R	I	III	III	C	I

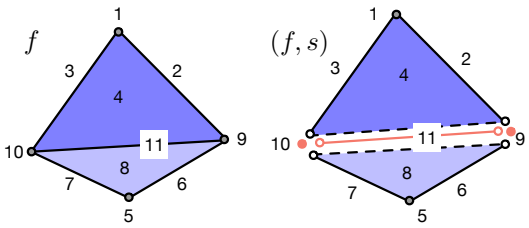


Figure 6 Example 5: split solid square. Every simplex is critical for (f, s).

Example 5: split solid square. As illustrated in Figure 6, the function f defined on a split solid square is not a discrete Morse function; there are three type I violators $f^{-1}(9)$, $f^{-1}(10)$, and $f^{-1}(11)$. Making these violators their own strata helps to convert f into a discrete stratified Morse function (f, s). In this example, all simplices are considered critical for (f, s). For instance, consider the open 2-simplex $f^{-1}(4)$, we have $L(f^{-1}(4)) = \{f^{-1}(11)\}$ and $U(f^{-1}(4)) = \emptyset$; with the stratification s in Figure 6 (right), $L_s(f^{-1}(4)) = \emptyset$ and so 4 is not a critical value for f but it is a critical value for (f, s). Since every simplex is critical for (f, s), there is no simplification to be done.

5 Proofs of main results

We now provide the proofs of our main results, Theorem 12, Theorem 13, and Theorem 14. To better illustrate our ideas, we construct "filtrations" by sublevel sets based upon the upside-down pentagon example (Figure 7).

5.1 Proof of Theorem 12

Proof. For simplicity, we suppose K is connected and f is 1-to-1; otherwise, based on the principle of simulation of simplicity [9], we may perturb f slightly without changing which cells are critical in K_a or K_b so that $f : K \to \mathbb{R}$ is 1-to-1. By partitioning $(a, b]$

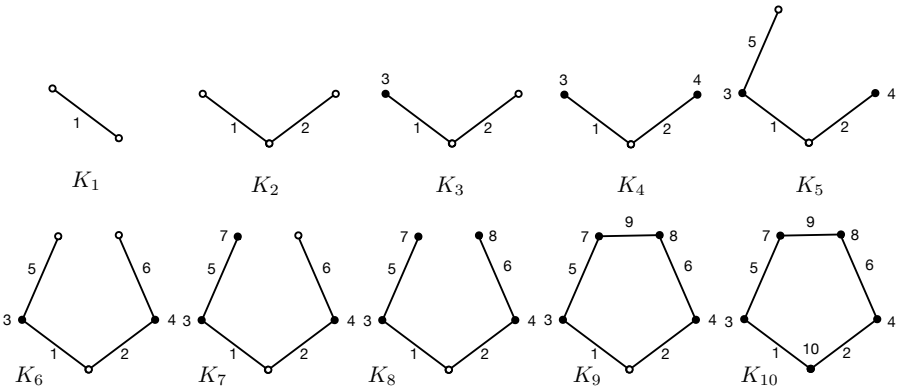


Figure 7 Example 1: upside-down pentagon. We show K_c as c increases from 1 to 10.

into smaller intervals if necessary, we may assume there is a single noncritical cell σ with $f(\sigma) \in (a, b]$. Since σ is noncritical, either (a) $|L_s(\sigma)| = 1$ and $|U_s(\sigma)| = 0$ or (b) $|U_s(\sigma)| = 1$ and $|L_s(\sigma)| = 0$.

Since case (a) requires that $p \geq 1$, we assume for now that is the case. There exists a single $\nu^{(p-1)} < \sigma$ with $f(\nu) > f(\sigma)$; such a $\nu \notin K_b$. Meanwhile, any other $(p-1)$-face $\tilde{\nu}^{(p-1)} < \sigma$ satisfies $f(\tilde{\nu}) < f(\sigma)$, implying $\tilde{\nu} \in K_a$. The set $\{\tilde{\nu}\}$ of such $\tilde{\nu}$ corresponds to the portion of the boundary of σ that lies in K_a, that is, $K_b = K_a \cup_{\{\tilde{\nu}\}} [\sigma)$, where $\tilde{\nu}$ are open faces of σ. Note that we use the half-closed simplex $[\sigma)$ to emphasize its boundary $\tilde{\nu}$ in K_a. We now apply Theorem 7 by setting $A = A' = Y = \{\tilde{\nu}\}$, $X = X' = K_a$, $Y' = [\sigma)$; i, i', φ_Y and f' the corresponding inclusions, and all other maps the identity. Since the diagram commutes and the pairs $(K_a, \{\tilde{\nu}\})$ and $(\sigma, \{\tilde{\nu}\})$ both satisfy the homotopy extension property, the maps $i = i'$ and φ_Y are cofibrations. It follows that $K_a = K_a \cup_{\{\tilde{\nu}\}} \{\tilde{\nu}\}$ and $K_b = K_a \cup_{\{\tilde{\nu}\}} [\sigma)$ are homotopy equivalent.

For case (b), σ has a single coface $\tau^{(p+1)} > \sigma$ with $f(\tau) < f(\sigma)$. Thus $\tau \in K_a$ and any other coface $\tilde{\tau} > \sigma$ must have a larger function value; that is, $\tilde{\tau} \notin K_b$. Denote by K'_a the set $K_a \setminus \tau$. Let $\{\omega\}$ denote the boundary of τ in K'_a. Then $K_a = K'_a \cup_{\{\omega\}} [\tau)$, and $K_b = K'_a \cup_{\{\omega\}} ([\tau) \cup \sigma)$ and σ is a free face of τ. We apply Theorem 7 by setting $A = A' = \{\omega\}$, $X = X' = K'_a$, $Y = [\tau)$, $Y' = [\tau) \cup \sigma$. The maps i, i', φ_Y and f' are inclusions and also cofibrations, while all other maps are the identity. Attaching σ to τ is clearly a homotopy equivalence and so we see that K_a and K_b are homotopy equivalent in this case as well.

Finally, it is clear that the above homotopy equivalence is stratum-preserving; in particular, the retracts associated with the inclusion/cofibration $\varphi_Y : \{\tilde{\nu}\} \to [\sigma)$ in case (a), and $\varphi_Y : [\tau) \to [\tau) \cup \sigma$ in case (b) are both completely contained within their own strata. Therefore, K_a and K_b are stratum-preserving homotopy equivalent. ◀

Examples of attaching regular simplices. Let's examine how this works in our upside-down pentagon example (Figure 7). Applying Theorem 12 going from K_4 to K_5, we attach the open simplex $f^{-1}(5)$ to its boundary in K_4, which consists of the single vertex $f^{-1}(4)$. The simplex $f^{-1}(5)$ is a regular simplex and so $K_4 \simeq K_5$. This is precisely case (a) in the proof of Theorem 12. Similarly, $K_6 \simeq K_7$, as $f^{-1}(6)$ is a regular simplex in its stratum, and this corresponds to case (b) in the proof of Theorem 12.

5.2 Proof of Theorem 13

Proof. Again, we may assume that f is 1-to-1. We may further assume that σ is the only simplex with a value between $(a, b]$ and prove that K_b is homotopy equivalent to $K_a \cup_{\dot{\sigma}|_{K_a}} \sigma$.

Based on the definition of K_c, since $f(\sigma) > a$, we know that $\sigma \cap K_a = \emptyset$. We now consider several cases. Let σ and (σ) denote open simplices and $[\sigma]$ denote the closure.

Case (a), suppose σ is not on the boundary of a stratum. Since σ is critical in its own stratum $s(\sigma)$, then for every $\nu^{(p-1)} < \sigma$ in the same stratum as σ (i.e. $s(\nu) = s(\sigma)$), we have $f(\nu) < f(\sigma)$, so that $f(\nu) < a$, which implies $\nu \in K_a$. In addition any such ν is not on the boundary of a stratum (otherwise σ would be part of the boundary). This means that all $(p-1)$-dimensional open faces of σ lying in $s(\sigma)$ are in K_a; this is precisely the boundary of σ in K_a, denoted $\dot{\sigma}|_{K_a}$. Therefore $K_b = K_a \cup_{\dot{\sigma}|_{K_a}} \sigma$.

Case (b), suppose σ is on the boundary of a stratum. There are two subcases: (i) σ is a violator in the sense of Definition 11 and therefore forms its own stratum; or (ii) σ is not a violator.

Case (b)(i), suppose σ is a type I violator; that is, globally $|U(\sigma)| \geq 2$. Then for any $\tau^{(p+1)} > \sigma$ in $U(\sigma)$ we have $f(\tau) \leq f(\sigma)$. If follows that $f(\tau) < a$, implying $\tau \in K_a$. Denote the set of such τ as $\{\tau\}$. Meanwhile, if $|L(\sigma)| = 0$, then for all $\nu^{(p-1)} < \sigma$ we have $f(\nu) < f(\sigma)$; that is, all the $(p-1)$-dimensional faces of σ are in K_a. Denote the set of such ν as $\{\nu\}$. The set $\{\nu\}$ is precisely $\dot{\sigma}|_{K_a}$. Therefore, $K_b = K_a \cup_{\dot{\sigma}|_{K_a}} \sigma$, where we are attaching σ along its whole boundary (which lies in K_a) and realizing it as a portion of $\dot{\tau}$ for each $\tau \in \{\tau\}$. On the other hand, if $|L(\sigma)| \neq 0$, let $\mu^{(p-1)} < \sigma$ denote any face of σ *not* in $L(\sigma)$. Again denote the set of such μ as $\{\mu\}$. The remaining $(p-1)$ faces $\nu < \sigma$ all lie in K_a; denote these by $\{\nu\}$. Note that $\{\nu\} = \dot{\sigma}|_{K_a}$. Then $\dot{\sigma} = \{\nu\} \cup \{\mu\}$ and $K_b = K_a \cup_{\dot{\sigma}|_{K_a}} \sigma$.

Now suppose σ is a type II violator, thus globally $|L(\sigma)| \geq 2$. The simplices $\nu^{(p-1)} < \sigma$ not in $L(\sigma)$ satisfy $f(\nu) < f(\sigma)$, thus such $\nu \in K_a$ form the (possibly empty) set $\{\nu\}$. The simplices $\tau^{(p+1)} > \sigma$ in $U(\sigma)$ satisfy $f(\tau) < f(\sigma)$ thus such $\tau \in K_a$ form the (possibly empty) set $\{\tau\}$. The set $\{\nu\}$ is precisely $\dot{\sigma}|_{K_a}$ and we again have $K_b = K_a \cup_{\dot{\sigma}|_{K_a}} \sigma$. Finally, suppose σ is a type III violator, the proof in this case is similar (and therefore omitted).

Case (b)(ii): σ is not a violator. Since σ is critical for a discrete stratified Morse function, it is either critical globally (i.e. $|U(\sigma)| = |L(\sigma)| = 0$) or locally (i.e. $|U_s(\sigma)| = |L_s(\sigma)| = 0$). Suppose σ is critical locally but not globally, meaning that either $|U(\sigma)| = 1, |L(\sigma)| = 0$, or $|U(\sigma)| = 0, |L(\sigma)| = 1$. If $|U(\sigma)| = 1$ and $|L(\sigma)| = 0$ globally, then $|U_s(\sigma)|$ becomes 0. If $\tau^{(p+1)} > \sigma$ is the unique element in $U(\sigma)$, then $f(\tau) < f(\sigma)$ and τ is in K_a. All cells $\nu^{(p-1)} < \sigma$ satisfy $f(\nu) < f(\sigma)$ and therefore are in K_a. The set $\{\nu\}$ again is precisely $\dot{\sigma}|_{K_a}$ and we have $K_b = K_a \cup_{\dot{\sigma}|_{K_a}} \sigma$, where we are attaching σ as a free face of τ. The cases when $|U(\sigma)| = 0, |L(\sigma)| = 1$, or $|U(\sigma)| = 0, |L(\sigma)| = 0$ are proved similarly.

In summary, when passing through a single, unique critical cell $\sigma^{(p)}$ with a function value in $(a, b]$, $K_b = K_a \cup_{\dot{\sigma}|_{K_a}} \sigma$. Since σ is homeomorphic to $e^{(p)}$, $K_b = K_a \cup_{\dot{e^{(p)}}|_{K_a}} e^{(p)}$. ◀

Examples of attaching critical simplices. Returning to the upside-down pentagon (Figure 7), we have a few critical cells, namely those with critical values $1, 2, 3, 4, 9$, and 10. Attaching $f^{-1}(2)$ to K_1, for example, changes the homotopy type, yielding a space with two connected components. Note that the boundary of this cell, restricted to K_1 is empty. When we attach $f^{-1}(9)$, we do so along its entire boundary (which lies in K_8), joining the two components together. Finally, attaching the vertex $f^{-1}(10)$ to K_9 changes the homotopy type yet again, yielding a circle.

5.3 Proof of Theorem 14

Proof. We assume K is connected. If f itself is a discrete Morse function, then there are no violators in K. The algorithm produces the trivial stratification $\mathcal{S} = \{K\}$ and since f is a discrete Morse function on the entire complex, the pair (f, s) trivially satisfies Definition 8.

If f is not a discrete Morse function, let $\mathcal{S} = \mathcal{V} \cup \{S_j\}$ denote the stratification produced by the algorithm. Since each violator α forms its own stratum $s(\alpha)$, the restriction of f to $s(\alpha)$ is trivially a discrete Morse function in which α is a critical simplex. It remains to show that the restriction of f to each S_j is a discrete Morse function.

If σ is a simplex in S_j, that is, $s(\sigma) = S_j$, consider the sets $U_s(\sigma)$ and $L_s(\sigma)$. Since σ is not a violator, the global sets $U(\sigma)$ and $L(\sigma)$ already satisfy the conditions required of an ordinary discrete Morse function. Restricting attention to the stratum $s(\sigma)$ can only reduce their size; that is, $|U_s(\sigma)| \leq |U(\sigma)|$ and $|L_s(\sigma)| \leq |L(\sigma)|$. It follows that the restriction of f to S_j is a discrete Morse function. ◀

Remark. When we restrict the function $f : K \to \mathbb{R}$ to one of the strata S_j, a non-violator σ that is regular globally (that is, σ forms a gradient pair with a unique simplex τ) may become a critical simplex for the restriction of f to S_j, e.g. $f^{-1}(3)$ in Figure 2 (top middle).

6 Discussion

In this paper we have identified a reasonable definition of a discrete stratified Morse function and demonstrated some of its fundamental properties. Many questions remain to be answered; we plan to address these in future work.

Relation to classical stratified Morse theory. An obvious question to ask is how our theory relates to the smooth case. Suppose X is a Whitney stratified space and $F : X \to \mathbb{R}$ is a stratified Morse function. One might ask the following: is there a triangulation K of X and a discrete stratified Morse function (f, s) on K that mirrors the behavior of F? That is, can we define a discrete stratified Morse function so that its critical simplices contain the critical points of the function F? This question has a positive answer in the setting of discrete Morse theory [3], so we expect the same to be true here as well.

Morse inequalities. Forman proved the discrete version of the Morse inequalities in [11]. Does our theory produce similar inequalities?

Discrete dynamics. Forman developed a more general theory of discrete vector fields [10] in which closed V-paths are allowed (analogous to recurrent dynamics). This yields a decomposition of a cell complex into pieces and an associated Lyapunov function (constant on the recurrent sets). This is not the same as a stratification, but it would be interesting to uncover any connections between our theory and this general theory. In particular, one might ask if there is some way to glue together the discrete Morse functions on each piece of a stratification into a global discrete vector field.

References

1. Madjid Allili, Tomasz Kaczynski, and Claudia Landi. Reducing complexes in multidimensional persistent homology theory. *Journal of Symbolic Computation*, 78:61–75, 2017.
2. Paul Bendich and John Harer. Persistent intersection homology. *Foundations of Computational Mathematics*, 11(3):305–336, 2011.
3. Bruno Benedetti. Smoothing discrete Morse theory. *Annali della Scuola Normale Superiore di Pisa*, 16(2):335–368, 2016.
4. Ronald Brown. *Topology and Groupoids*. www.groupoids.org, 2006.
5. Olaf Delgado-Friedrichs, Vanessa Robins, and Adrian Sheppard. Skeletonization and partitioning of digital images using discrete Morse theory. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):654–666, 2015.

6 Pawel Dlotko and Hubert Wagner. Computing homology and persistent homology using iterated Morse decomposition, 2012. URL: https://arxiv.org/abs/1210.1429.

7 Herbert Edelsbrunner, John Harer, Vijay Natarajan, and Valerio Pascucci. Morse-Smale complexes for piece-wise linear 3-manifolds. *ACM Symposium on Computational Geometry*, pages 361–370, 2003.

8 Herbert Edelsbrunner, John Harer, and Afra J. Zomorodian. Hierarchical Morse-Smale complexes for piecewise linear 2-manifolds. *Discrete & Computational Geometry*, 30(87-107), 2003.

9 Herbert Edelsbrunner and Ernst Peter Mücke. Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics*, 9(1):66–104, 1990.

10 Robin Forman. Combinatorial vector fields and dynamical systems. *Mathematische Zeitschrift*, 228(4):629–681, 1998.

11 Robin Forman. Morse theory for cell complexes. *Advances in Mathematics*, 134:90–145, 1998.

12 Robin Forman. Combinatorial differential topology and geometry. *New Perspectives in Geometric Combinatorics*, 38:177–206, 1999.

13 Robin Forman. A user's guide to discrete Morse theory. *Séminaire Lotharingien de Combinatoire*, 48, 2002.

14 Greg Friedman. Stratified fibrations and the intersection homology of the regular neighborhoods of bottom strata. *Topology and its Applications*, 134(2):69–109, 2003.

15 Mark Goresky and Robert MacPherson. *Stratified Morse Theory.* Springer-Verlag, 1988.

16 David Günther, Jan Reininghaus, Hans-Peter Seidel, and Tino Weinkauf. Notes on the simplification of the Morse-Smale complex. In Peer-Timo Bremer, Ingrid Hotz, Valerio Pascucci, and Ronald Peikert, editors, *Topological Methods in Data Analysis and Visualization III*, pages 135–150. Springer International Publishing, 2014.

17 Attila Gyulassy, Peer-Timo Bremer, Bernd Hamann, and Valerio Pascucci. A practical approach to Morse-Smale complex computation: Scalability and generality. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1619–1626, 2008.

18 Henry King, Kevin Knudson, and Neža Mramor. Generating discrete Morse functions from point data. *Experimental Mathematics*, 14:435–444, 2005.

19 Manfred Knebusch. Semialgebraic topology in the last ten years. In Michel Coste, Louis Mahe, and Marie-Francoise Roy, editors, *Real Algebraic Geometry*, 1991.

20 Kevin Knudson and Bei Wang. Discrete stratified Morse theory: A user's guide. 2018. URL: https://arxiv.org/abs/1801.03183.

21 Konstantin Mischaikow and Vidit Nanda. Morse theory for filtrations and efficient computation of persistent homology. *Discrete & Computational Geometry*, 50(2):330–353, 2013.

22 James R. Munkres. *Elements of algebraic topology.* Addison-Wesley, Redwood City, California, 1984.

23 Jan Reininghaus. *Computational Discrete Morse Theory.* PhD thesis, Zuse Institut Berlin (ZIB), 2012.

24 Jan Reininghaus, Jens Kasten, Tino Weinkauf, and Ingrid Hotz. Efficient computation of combinatorial feature flow fields. *IEEE Transactions on Visualization and Computer Graphics*, 18(9):1563–1573, 2011.

25 Vanessa Robins, Peter John Wood, and Adrian P. Sheppard. Theory and algorithms for constructing discrete Morse complexes from grayscale digital images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1646–1658, 2011.

26 Tammo tom Dieck. *Algebraic Topology.* European Mathematical Society, 2008.

An Optimal Algorithm to Compute the Inverse Beacon Attraction Region

Irina Kostitsyna
Department of Mathematics and Computer Science, TU Eindhoven, the Netherlands
i.kostitsyna@tue.nl

Bahram Kouhestani
School of Computing, Queen's University, Kingston, Ontario, Canada
kouhesta@cs.queensu.ca

Stefan Langerman[1]
Département d'informatique, Université Libre de Bruxelles, Brussels, Belgium
stefan.langerman@ulb.ac.be

David Rappaport
School of Computing, Queen's University, Kingston, Ontario, Canada
daver@cs.queensu.ca

Abstract

The *beacon model* is a recent paradigm for guiding the trajectory of messages or small robotic agents in complex environments. A *beacon* is a fixed point with an attraction pull that can move points within a given polygon. Points move greedily towards a beacon: if unobstructed, they move along a straight line to the beacon, and otherwise they slide on the edges of the polygon. The Euclidean distance from a moving point to a beacon is monotonically decreasing. A given beacon *attracts* a point if the point eventually reaches the beacon.

The problem of attracting all points within a polygon with a set of beacons can be viewed as a variation of the art gallery problem. Unlike most variations, the beacon attraction has the intriguing property of being asymmetric, leading to separate definitions of *attraction region* and *inverse attraction region*. The attraction region of a beacon is the set of points that it attracts. It is connected and can be computed in linear time for simple polygons. By contrast, it is known that the inverse attraction region of a point – the set of beacon positions that attract it – could have $\Omega(n)$ disjoint connected components.

In this paper, we prove that, in spite of this, the total complexity of the inverse attraction region of a point in a simple polygon is linear, and present a $O(n \log n)$ time algorithm to construct it. This improves upon the best previous algorithm which required $O(n^3)$ time and $O(n^2)$ space. Furthermore we prove a matching $\Omega(n \log n)$ lower bound for this task in the algebraic computation tree model of computation, even if the polygon is monotone.

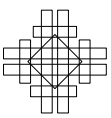
2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases beacon attraction, inverse attraction region, algorithm, optimal

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.55

Related Version A full version of this paper is available at [12], `http://arxiv.org/abs/1803.05946`.

[1] Directeur de Recherches du F.R.S.-FNRS.



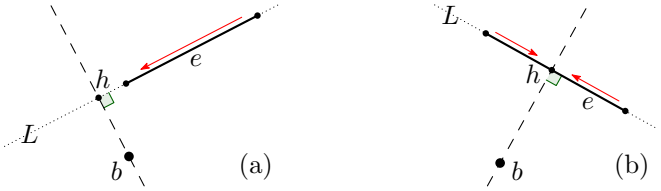


Figure 1 Points on an edge e slide towards the orthogonal projection h of the beacon on the supporting line of e.

1 Introduction

Consider a dense network of sensors. In practice, it is common that routing between two nodes in the network is performed by greedy geographical routing, where a node sends the message to its closest neighbor (by Euclidean distance) to the destination [11]. Depending on the geometry of the network, greedy routing may not be successful between all pairs of nodes. Thus, it is essential to determine nodes of the network for which this type of routing works. In particular, given a node in the network, it is important to compute all nodes that can successfully send a message to (or receive a message from) the input node. Motivated by this application Biro *et al.* [3] introduced the beacon routing model.

Let P be a simple polygon with n vertices. A *beacon* b is a point in P that can induce an attraction pull towards itself within P. The attraction of b causes points in P to move towards b as long as their Euclidean distance is maximally decreasing. As a result, a point p moves along the ray $\overrightarrow{pb}$ until it either reaches b or an edge of P. In the latter case, p slides on the edge towards h, the orthogonal projection of b on the supporting line of the edge (Figure 1). Note that among all points on the supporting line of the edge, h has the minimum Euclidean distance to b.

We say b *attracts* p, if p eventually reaches b. Interestingly, beacon attraction is not symmetric. The *attraction region* of b, denoted by $AR(b)$, is the set of all points in P that b attracts[2]. The *inverse attraction region* of a point p, denoted by $IAR(p)$, is the set of all beacon positions in P that can attract p.

The study of beacon attraction problems in a geometric domain, initiated by Biro *et al.* [3], finds its root in sensor networks, where the limited capabilities of sensors makes it crucial to design simple mechanisms for guiding their motion and communication. For instance, the beacon model can be used to represent the trajectory of small robotic agents in a polygonal domain, or that of messages in a dense sensor network. Using greedy routing, the trajectory of a robot (or a message) from a sender to a receiver closely follows the attraction trajectory of a point (the sender) towards a beacon (the receiver). However, greedy routing may not be successful between all pairs of nodes. Thus, it is essential to characterize for which pairs of nodes of the network for which this type of routing works. In particular, given a single node, it is important to compute the set of nodes that it can successfully receive messages from (its attraction region), and the set of node that it can successfully send messages to (its inverse attraction region).

In 2013, Biro *et al.* [5] showed that the attraction region $AR(b)$ of a beacon b in a simple polygon P is simple and connected, and presented a linear time algorithm to compute $AR(b)$.

[2] We consider the attraction region to be closed, *i.e.*, b attracts all points on the boundary of $AR(b)$.

Computing the inverse attraction region has proven to be more challenging. It is known [5] that the inverse attraction region $IAR(p)$ of a point p is not necessarily connected and can have $\Theta(n)$ connected components. Kouhestani *et al.* [14] presented an algorithm to compute $IAR(p)$ in $O(n^3)$ time and $O(n^2)$ space. In the special cases of monotone and terrain polygons, they showed improved algorithms with running times $O(n \log n)$ and $O(n)$ respectively.

In this paper, we prove that, in spite of not being connected, the inverse attraction region $IAR(p)$ always has total complexity[3] $O(n)$. Using this fact, we present the first optimal $O(n \log n)$ time algorithm for computing $IAR(p)$ for any simple polygon P, improving upon the previous best known $O(n^3)$ time algorithm. Since this task is at the heart of other algorithms for solving beacon routing problems, this improves the time complexity of several previously known algorithms such as approximating minimum beacon paths and computing the weak attraction region of a region [5].

To prove the optimality of our algorithm, we show an $\Omega(n \log n)$ lower bound in the algebraic computation tree model and in the bounded degree algebraic decision tree model, even in the case when the polygon is monotone.

Due to space limitations some of the proofs are omitted and can be found in the full version of this paper [12].

Related work

Greedy routing has been studied extensively in the literature of sensor network as a local (and therefore inexpensive) protocol for message sending. As a result, many applications in wireless and sensor networks utilize greedy routing to choose the next hop in their message sending protocol [10]. In the geometric domain, greedy routing has been studied in both discrete and continuous spaces. Bose *et al.* [6] studied routing problems in ad hoc wireless networks modeled as unit graphs and Kermarrec and Tan [11] presented an approximation algorithm to decompose a polygon into minimum number of routable regions, *i.e.*, regions in which greedy routing always works. Beacon routing, discussed in this paper, is essentially greedy routing in a polygonal environment representing an infinitely dense sensor network.

Several geometric problems related to the beacon model have been studied in recent years. Biro *et al.* [3] studied the minimum number of beacons necessary to successfully route between any pair of points in a simple n-gon P. This can be viewed as a variant of the art gallery problem, where one wants to find the minimum number of beacons whose attraction regions cover P. They proved that $\lceil \frac{n}{2} \rceil$ beacons are sometimes necessary and always sufficient, and showed that finding a minimum cardinality set of beacons to cover a simple polygon is NP-hard. For polygons with holes, Biro *et al.* [4] showed that $\lceil \frac{n}{2} \rceil - h - 1$ beacons are sometimes necessary and $\lceil \frac{n}{2} \rceil + h - 1$ beacons are always sufficient to guard a polygon with h holes. Combinatorial results on the use of beacons in orthogonal polygons have been studied by Bae *et al.* [1] and by Shermer [17]. Biro *et al.* [5] presented a polynomial time algorithm for routing between two fixed points using a discrete set of candidate beacons in a simple polygon and gave a 2-approximation algorithm where the beacons are placed with no restrictions. Kouhestani *et al.* [15] give an $O(n \log n)$ time algorithm for beacon routing in a 1.5D polygonal terrain.

Kouhestani *et al.* [13] showed that the length of a successful beacon trajectory is less than $\sqrt{2}$ times the length of a shortest (geodesic) path. In contrast, if the polygon has internal holes then the length of a successful beacon trajectory may be unbounded.

[3] Total number of vertices and edges of all connected components.

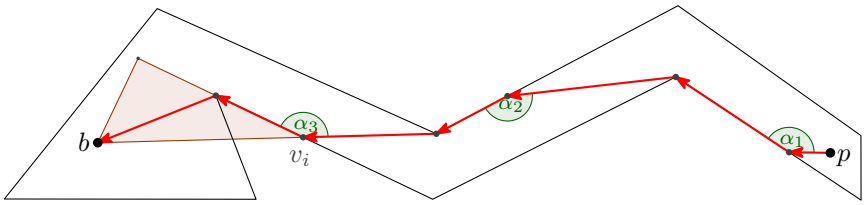


Figure 2 The angle between a straight movement towards the beacon and the following slide movement is always greater than $\pi/2$.

2 Preliminaries

A *dead point* $d \neq b$ is defined as a point that remains stationary in the attraction pull of b. The set of all points in P that eventually reach (and stay) on d is called the *dead region* of b with respect to d. A *split edge* is defined as the boundary between two dead regions, or a dead region and $AR(b)$. In the latter case, we call the split edge a *separation edge*.

If beacon b attracts a point p, we use the term *attraction trajectory*, denoted by $AT(p, b)$, to indicate the movement path of a point p from its original location to b. The attraction trajectory alternates between a straight movement towards the beacon (a *pull edge*) and a sequence of consecutive sliding movements (*slide edges*), see Figure 2.

▶ **Lemma 1.** *Consider the attraction trajectory $AT(p, b)$ of a point p attracted by beacon b. Let α_i denote the angle between the i-th pull edge and the next slide edge on $AT(p, b)$. Then α_i is greater than $\pi/2$.*

Note that, similarly, the angle between the i-th pull edge and the previous slide edge is also greater than $\pi/2$.

Let r be a reflex vertex of P with adjacent edges e_1 and e_2. Let H_1 be the half-plane orthogonal to e_1 at r, that contains e_1. Let H_2 be the half-plane orthogonal to e_2 at r, that contains e_2. The *deadwedge* of r (deadwedge(r)) is defined as $H_1 \cap H_2$ (Figure 3). Let b be a beacon in the deadwedge of r. Let ρ be the ray from r in the direction $\overrightarrow{br}$ and let s be the line segment between r and the first intersection of ρ with the boundary of P. Note that in the attraction of b, points on different sides of s have different destinations. Thus, s is a split edge for b. We say r *introduces* the split edge s for b to show this occurrence. Kouhestani *et al.* [14] proved the following lemma.

▶ **Lemma 2** (Kouhestani *et al.* [14])**.** *A reflex vertex r introduces a split edge for the beacon b if and only if b is inside the deadwedge of r.*

Let p and q be two points in a polygon P. We use $\overline{pq}$ to denote the straight-line segment between these points. Denote the shortest path between p and q in P (the geodesic path)

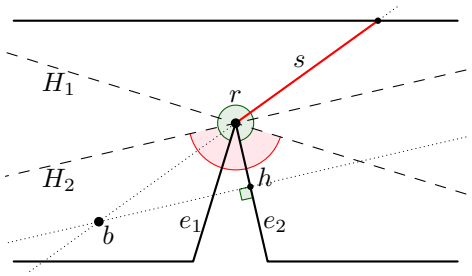


Figure 3 The deadwedge of r is shown by the red angle.

as $SP(p, q)$. The union of shortest paths from p to all vertices of P is called the *shortest path tree* of p, and can be computed in linear time [9] when P is a simple polygon. In our problem, we are only interested in shortest paths from p to reflex vertices of P. Therefore, we delete all convex vertices and their adjacent edges in the shortest path tree of p to obtain the *pruned shortest path tree* of p, denoted by $SPT_r(p)$.

A *shortest path map* for a given point p, denoted as $SPM(p)$, is a subdivision of P into regions such that shortest paths from p to all the points inside the same region pass through the same set of vertices of P [16]. Typically, shortest path maps are considered in the context of polygons with holes, where the subdivision represents grouping of the shortest paths of the same topology, and the regions may have curved boundaries. In the case of a simple polygon, the boundaries of $SPM(p)$ are straight-line segments and consist solely of the edges of P and extensions of the edges of $SPT_r(p)$. If a triangulation of P is given, it can be computed in linear time [9].

▶ **Lemma 3.** *During the movement of p on its beacon trajectory, the shortest path distance of p away from its original location monotonically increases.*

3 The structure of inverse attraction regions

The $O(n^3)$ time algorithm of Kouhestani *et al.* [14] to compute the inverse attraction region of a point p in a simple polygon P constructs a line arrangement A with quadratic complexity that partitions P into regions, such that, either all or none of the points in a region attract p. Arrangement A, contains three types of lines:

1. Supporting lines of the deadwedge for each reflex vertex of P,
2. Supporting lines of edges of $SPT_r(p)$,
3. Supporting lines of edges of P.

▶ **Lemma 4** (Kouhestani *et al.* [14])**.** *The boundary edges of $IAR(p)$ lie on the lines of arrangement A.*

Let $\overline{uv}$ be an edge of $SPT_r(p)$, where $u = parent(v)$. We associate three lines of the arrangement A to $\overline{uv}$: supporting line of $\overline{uv}$ and the two supporting lines of the deadwedge of v. By focusing on the edge $\overline{uv}$, we study the local effect of the reflex vertex v on $IAR(p)$, and we show that:

1. Exactly one of the associated lines to $\overline{uv}$ may contribute to the boundary of $IAR(p)$. We call this line the *effective associated line* of $\overline{uv}$ (Figure 4).
2. The effect of v on the inverse attraction region can be represented by at most two half-planes, which we call the *constraining half-planes* of $\overline{uv}$. These half-planes are bounded by the effective associated line of $\overline{uv}$.
3. Each constraining half-plane has a *domain*, which is a subpolygon of P that it affects. The points of the constraining half-plane that are inside the domain subpolygon cannot attract p (see the next section).

Our algorithm to compute the inverse attraction region uses $SPM(p)$. For each region of $SPM(p)$, we compute the set of constraining half-planes with their domain subpolygons containing the region. Then, we discard points of the region that cannot attract p by locating points which belong to at least one of these constraining half-planes.

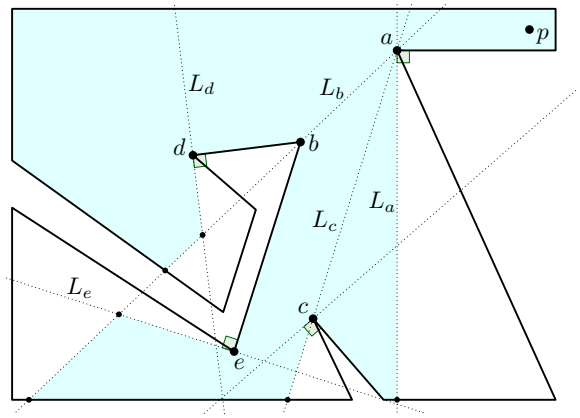


Figure 4 An example of an inverse attraction region with effective associated lines to each reflex vertex. Points in the colored region attract p. Here L_a, L_b, L_c, L_d and L_e are respectively the associated lines of the reflex vertices a, b, c, d and e.

Constraining half-planes

Let $\overline{uv}$ be an edge of $SPT_r(p)$, where $u = parent(v)$. We extend $\overline{uv}$ from u until we reach w, the first intersection with the boundary of P. Segment $\overline{uw}$ partitions P into two subpolygons. Let P_p be the subpolygon that contains p. Any path from p to any point in $P \setminus P_p$ passes through $\overline{uw}$. Thus a beacon outside of P_p that attracts p, must be able to attract at least one point on the line segment $\overline{uw}$. In order to determine the local attraction behaviour caused by the vertex v, and to find the effective line associated to $\overline{uv}$, we focus on the attraction pull on the points of $\overline{uw}$ (particularly the vertex u) rather than p. By doing so we detect points that cannot attract u, or any point on $\overline{uw}$, and mark them as points that cannot attract p. In other words, for each edge $\overline{uv} \in SPT_r(p)$ we detect a set of points in P that cannot attract u locally due to v. The attraction of these beacons either causes u to move to a wrong subpolygon, or their attraction cannot move u past v (see the following two cases for details). Later in Theorem 8, we show that this suffices to detect all points that cannot attract p.

Let e_1 and e_2 be the edges incident to v. Let H_1 be the half-plane, defined by a line orthogonal to e_1 passing through v, which contains e_1, and let H_2 be the half-plane, defined by a line orthogonal to e_2 passing through v, which contains e_2. Depending on whether u is in $H_1 \cup H_2$, we consider two cases:

Case 1. Vertex u is not in $H_1 \cup H_2$ (Figure 5). We show that in this case the supporting line of $\overline{uv}$ is the only line associated to v that may contribute to the boundary of $IAR(p)$, *i.e.*, it is the effective line associated to $\overline{uv}$. Let q be an arbitrary point on the open edge e_1. As u is not in $H_1 \cup H_2$, the angle between the line segments $\overline{uq}$ and $\overline{qv}$ is less than $\pi/2$. Consider an arbitrary attraction trajectory that moves u straight towards q. By Lemma 1, any slide movement of this attraction trajectory on the edge e_1 moves away from v. Now consider q to be on the edge e_2. Similarly any slide on the edge e_2 moves away from v. Thus, the line segment $\overline{uv}$ can only be crossed once in an attraction trajectory of u (and, similarly, of any other point on the line segment $\overline{uw}$). Note that this crossing movement happens via a pull edge. We use this observation to detect a set of points that do not attract u and thus do not attract p.

Now consider the supporting line L of the edge $\overline{uv}$. As u is not in $H_1 \cup H_2$, L partitions

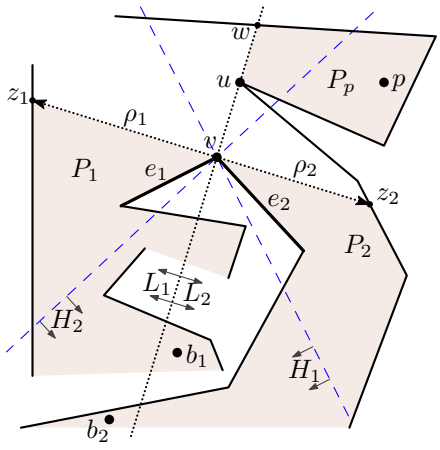


Figure 5 Vertex $u \notin H_1 \cup H_2$. Subpolygon P_2 is the domain of the constraining half-plane H_1, and P_1 is the domain of the constraining half-plane H_2.

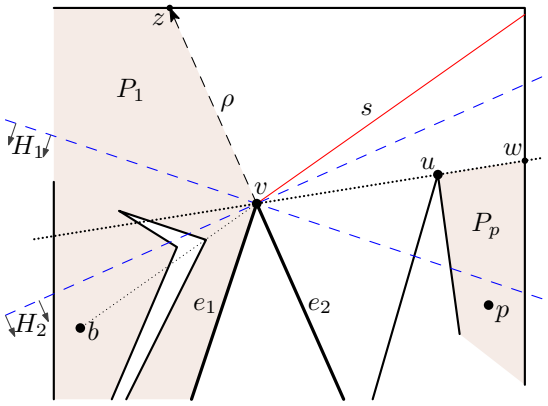


Figure 6 Vertex $u \in H_1 \cup H_2$. Subpolygon P_1 is the domain of the constraining half-plane H_2.

the plane into two half-planes L_1 containing the edge e_1, and L_2 containing the edge e_2. Without loss of generality, assume that the parent of u in $SPT_r(p)$ lies inside L_2 (refer to Figure 5). Recall that $\overline{uw}$ partitions P into two subpolygons, and P_p is the subpolygon containing p. We define subpolygons P_1 and P_2 as follows. Let ρ_1 be the ray originating at v, perpendicular to L in L_1, and let z_1 be the first intersection point of ρ_1 with the boundary of P. Define P_1 as the subpolygon of P induced by $\overline{vz_1}$ that contains the edge e_1. Similarly, let ρ_2 be the ray originating at v, perpendicular to L inside L_2, and let z_2 be the first intersection point of ρ_2 with the boundary of P. Define P_2 as the subpolygon of P induced by $\overline{vz_2}$ that contains the edge e_2. We provide the details of the following two lemmas in the full version of this paper [12].

▶ **Lemma 5.** *No point in $P_1 \cap L_2$ can attract p.*

▶ **Lemma 6.** *No point in $P_2 \cap L_1$ can attract p.*

In summary, in case 1, the effect of $\overline{uv}$ is expressed by two half-planes: L_2, affecting the subpolygon P_1, and L_1, affecting the subpolygon P_2. We call L_1 and L_2 the *constraining half-planes* of $\overline{uv}$, and we call P_1 and P_2 the *domain* of the constraining half-planes L_2 and L_1, respectively. Furthermore, we call $P_1 \cap L_2$ and $P_2 \cap L_1$ the *constraining regions* of $\overline{uv}$. Later we show that L is the only effective line associated to $\overline{uv}$.

Case 2. Vertex u is in $H_1 \cup H_2$ (refer to Figure 6). Without loss of generality assume u can see part of the edge e_2. Similar to the previous case, we define the subpolygon P_p; let w be the first intersection of the ray $\overrightarrow{vu}$ with the boundary of P. Note that $\overline{uw}$ partitions P into two subpolygons. Let P_p be the subpolygon containing p. Now let ρ be the ray originating at v, along the extension of edge e_2. Let z be the first intersection of ρ with the boundary of P. We use P_1 to denote the subpolygon induced by $\overline{vz}$ that contains e_1. We detect points in P_1 that cannot move u (past v) into P_1.

▶ **Lemma 7.** *No point in $P_1 \cap H_2$ can attract p.*

In summary, in case 2, the effect of $\overline{uv}$ on $IAR(p)$ can be expressed by the half-plane H_2. We call H_2 the *constraining half-plane* of $\overline{uv}$, P_1 the *domain* of H_2 and we call $P_1 \cap H_2$ the

constraining region of $\overline{uv}$. Later we show that the supporting line of H_2 is the only effective line associated to v.

By combining these two cases, we prove the following theorem.

▶ **Theorem 8.** *A beacon b can attract a point p if and only if b is not in a constraining region of any edge of $SPT_r(p)$.*

Proof. By Lemmas 5, 6 and 7, if b is in the constraining region of an edge $\overline{uv} \in SPT_r(p)$ then it does not attract p.

Now let b be a point that cannot attract p. We will show that b is in the constraining region of at least one edge of $SPT_r(p)$. Let s be the separation edge of $AR(b)$ such that b and p are in different subpolygons induced by s (see, for example, Figure 6). Note that as the attraction region of a beacon is connected [2], there is exactly one such separation edge. Let v be the reflex vertex that introduces s and let u be the parent of v in $SPT_r(p)$. By Lemma 2, b is in the deadwedge of v. In addition, as the attraction region of a beacon is connected, b attracts v. We claim that b is in a constraining region of the edge $\overline{uv} \in SPT_r(p)$. First, we show that b cannot attract u. Consider $SP(p, u)$, the shortest path from p to u. If $SP(p, u)$ crosses s at some point q then u cannot be the parent of v in $SPT_r(p)$, because we can reach v with a shorter path by following $SP(p, u)$ from p to q and then reaching v from q. Therefore, $SP(p, u)$ does not cross s, so p and u are in the same subpolygon of P induced by s. As b does not attract p, we conclude that b does not attract u.

Consider the two cases: u is in $H_1 \cup H_2$ or not. We show that in each case, b is in a constraining region of $\overline{uv}$.

Case 1. Vertex u is not in $H_1 \cup H_2$ (refer to Figure 5). Let L be the supporting line of $\overline{uv}$, and similar to the previous case analysis let L_1 and L_2 be the constraining half-planes, and let P_1 and P_2 be the domains of L_2 and L_1, respectively. Without loss of generality, assume that b is in the half-plane L_2. We show that then b belongs to P_1.

As $b \in L_2$, the separation edge s extends from v into L_1, *i.e.*, $s \in L_1$. Then the point p and subpolygon P_2 lie on one side of s, and subpolygon P_1 lies on the other side of s. As beacon b does not attract p, the point p and the beacon b lie on different sides of s, and thus the beacon b and subpolygon P_1 lie on the same side of s.

We will show now that indeed $b \in P_1$. Beacon b attracts v and is in the deadwedge of v. Thus, in the attraction of b, v will enter P_1 via a slide move. We claim that v cannot leave P_1 afterwards. Consider the supporting line of ρ_1 which is a line orthogonal to $\overline{uv}$ at v. As u is not in $H_1 \cup H_2$, and the deadwedge of v is equal to $H_1 \cap H_2$, the deadwedge of v completely lies to one side of the supporting line. Therefore, in the attraction of v by any beacon inside the deadwedge of v, any point $q \neq v$ on $\overline{vz_1}$ moves straight towards the beacon along the ray $\overrightarrow{qb}$. In other words, in the attraction pull of b no point inside P_1 can leave P_1. Therefore, $b \in P_1$ and thus $b \in P_1 \cap L_2$. By definition, b belongs to a constraining region of $\overline{uv}$.

Case 2. Vertex u is in $H_1 \cup H_2$ (refer to Figure 6). Without loss of generality let $u \in H_2$. Consider the separation edge s. As the beacon b does not attract u, they lie on the opposite sides of s. As b is in the deadwedge of v, it is also in H_2, the constraining half-plane of $\overline{uv}$. Similar to the previous case, as b attracts v, $AT(v, b)$ never crosses ρ to leave P_1 and therefore, b is in P_1. Thus, $b \in P_1 \cap H_2$ and it belongs to the constraining region of $\overline{uv}$. ◀

▶ **Corollary 9.** *Consider the edge $\overline{uv} \in SPT_r(p)$. If u is not in $H_1 \cup H_2$ (case 1), then among three associated lines to $\overline{uv}$ only the supporting line of $\overline{uv}$ may contribute to the boundary of $IAR(p)$. If u is in $H_1 \cup H_2$ (case 2), then among three associated lines to $\overline{uv}$ only the*

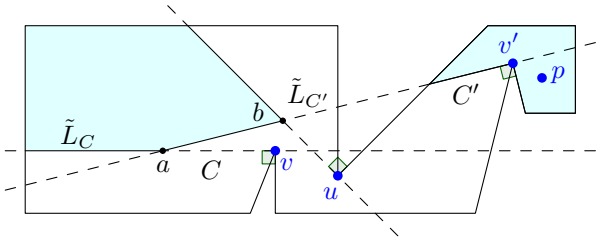


Figure 7 The charging scheme: vertex a is charged to the constraining half-plane C of vertex v. The inverse attraction region of p is the shaded region.

supporting line of H_2 may contribute to the boundary of $IAR(p)$, where H_2 is the half-plane orthogonal to the incident edge of v that u can partially see.

4 The complexity of the inverse attraction region

In this section we show that in a simple polygon P the complexity of $IAR(p)$ is linear with respect to the size of P.

We classify the vertices of the inverse attraction region into two groups: 1) vertices that are on the boundary of P, and 2) internal vertices. We claim that there are at most a linear number of vertices in each group. Throughout this section, without loss of generality, we assume that no two constraining half-planes of different edges of the shortest path tree are co-linear. Note that we can reach such a configuration with a small perturbation of the input points, which may just add to the number of vertices of $IAR(p)$.

Biro [2] showed that the inverse attraction region of a point in a simple polygon P is convex with respect to[4] P. Therefore, we have at most two vertices of $IAR(p)$ on each edge of P, and thus there are at most a linear number of vertices in the first group.

We use the following property of the attraction trajectory to count the number of vertices in group 2.

▶ **Lemma 10.** *Let L be the effective line associated to the edge $\overline{uv} \in SPT_r(p)$, where $u = parent(v)$. Let b be a beacon on $L \cap deadwedge(v)$ that attracts p. Then the attraction trajectory of p passes through both u and v.*

Next we define an ordering on the constraining half-planes. Let C be a constraining half-plane of the edge $\overline{uv} \in SPT_r(p)$ ($u = parent(v)$), and let C' be a constraining half-plane of the edge $\overline{u'v'} \in SPT_r(p)$ ($u' = parent(v')$). We say $C \leq C'$ if and only if $|SP(p, v)| \leq |SP(p, v')|$ (refer to Figure 7).

We use a charging scheme to count the number of internal vertices. An internal vertex resulting from the intersection of two constraining half-planes C and C' is charged to C' if $C \leq C'$, otherwise it is charged to C. In the remaining of this section, we show that each constraining half-plane is charged at most twice. Let P_C and P'_C denote the constraining regions related to C and C', respectively. And let L_C and $L_{C'}$ denote the supporting lines of C and C', respectively. In the previous section we showed that the line segments $L_C \cap P_C$ are the only parts of L_C that may contribute to the boundary of $IAR(p)$. Let $s \in L_C \cap P_C$ be a segment outside of the deadwedge of v. The next lemma shows that s does not appear on the boundary of $IAR(p)$, and we can ignore s when counting the internal vertices of $IAR(p)$.

[4] A subpolygon $Q \subseteq P$ is *convex with respect to* the polygon P if the line segment connecting two arbitrary points of Q either completely lies in Q or intersects P.

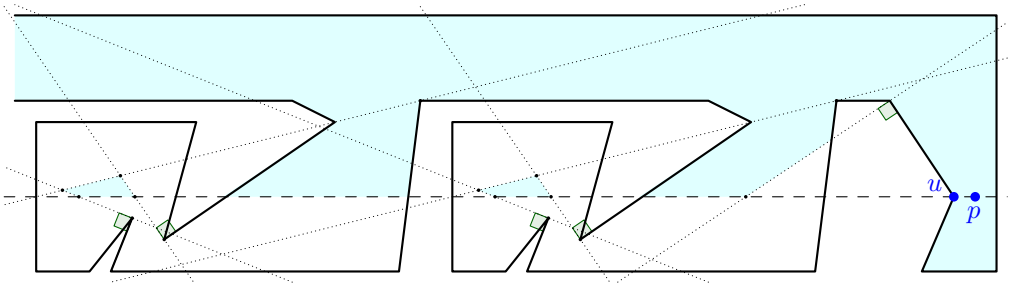


Figure 8 A constraining half-plane may contribute $O(n)$ vertices of group 2 to the inverse attraction region. Here the inverse attraction region of p is colored.

▶ **Lemma 11.** *Let $s \in L_C \cap P_C$ be a segment outside of the deadwedge of v. Then s (or a part of s with a non-zero length) does not appear on the boundary of $IAR(p)$.*

We define $\tilde{L}_C = L_C \cap P_C \cap \text{deadwedge}(v)$ and $\tilde{L}_{C'} = L_{C'} \cap P_{C'} \cap \text{deadwedge}(v')$. By Lemma 11, $\tilde{L}_C$ and $\tilde{L}_{C'}$ are the subset of L_C and $L_{C'}$ that may appear on the boundary of $IAR(p)$, therefore, the intersection points of all $\tilde{L}_C$ and $\tilde{L}_{C'}$ are the only possible locations for internal vertices of $IAR(p)$. Consider an internal vertex a resulting from the intersection of $\tilde{L}_C$ and $\tilde{L}_{C'}$.

▶ **Lemma 12.** *Let $a = \tilde{L}_C \cap \tilde{L}_{C'}$ be an internal vertex of $IAR(p)$ and let $C' \leq C$ (Figure 7). Then all points on $\tilde{L}_C$ are in the domain of C'.*

We charge a to C if $C' \leq C$, otherwise we charge it to C'. Assume a is charged to C. By Lemma 12, all points on $\tilde{L}_C$ to one side of a belong to the domain of C' and therefore are in C'. Thus, C cannot contribute any other internal vertices to this side of a. This implies that C can be charged at most twice (once from each end) and as there are a linear number of constraining half-planes, we have at most a linear number of vertices of group 2, and we have the following theorem.

▶ **Theorem 13.** *The inverse attraction region of a point p has linear complexity in a simple polygon.*

Note that, as illustrated in Figure 8, a constraining half-plane may contribute many vertices of group 2 to the inverse attraction region, but nevertheless it is charged at most twice.

5 Computing the inverse attraction region

In this section we show how to compute the inverse attraction region of a point inside a simple polygon in $O(n \log n)$ time.

Let region R_i of the shortest path map $SPM(p)$ consist of all points t such that the last segment of the shortest path from p to t is $\overline{v_i t}$ (Figure 9). Vertex v_i is called the *base* of R_i. Extend the edge of $SPT_r(p)$ ending at v_i until the first intersection z_i with the boundary of P. Call the segment $w_i = \overline{v_i z_i}$ a *window*, and point z_i – the *end* of the window; window w_i is a boundary segment of R_i.

We will construct a part of the inverse attraction region of p inside each region of the shortest path map $SPM(p)$ independently. A point in a region of $SPM(p)$ attracts p only if its attraction can move p into the region through the corresponding window.

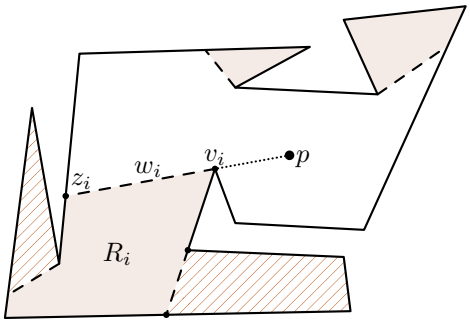


Figure 9 R_i is a region of $SPM(p)$ with base v_i. Segment w_i is the window, and z_i – its end.

▶ **Lemma 14.** *Let R_i be a region of $SPM(p)$ with a base vertex v_i. If v_i lies in some domain subpolygon P_e, then any point t in R_i lies in P_e.*

Let R_i be a region of $SPM(p)$ with a base vertex v_i, and let $\mathcal{H}_i$ be the set of all constraining half-planes corresponding to the domain subpolygons that contain the point v_i. Denote $Free_i$ to be the intersection of the complements of the half-planes in $\mathcal{H}_i$. Note, that $Free_i$ is a convex set. In the following lemma we show that $Free_i \cap R_i$ is exactly the set of points inside R_i that can attract p.

▶ **Lemma 15.** *The set of points in R_i that attract p is $Free_i \cap R_i$.*

This results in the following algorithm for computing the inverse attraction region of p. We compute the constraining half-planes of every edge of $SPT_r(p)$ of p and the corresponding domain subpolygons. Then, for every region R_i of the shortest path map of p, we compute the free region $Free_i$, where v_i is the base vertex of the region; and we add the intersection of R_i and $Free_i$ to the inverse attraction region of p. The pseudocode is presented in Algorithm 1.

Rather than computing each free space from scratch, we can compute and update free spaces using the data structure of Brodal and Jacob [7]. Their data structure allows to dynamically maintain the convex hull of a set of points and supports insertions and deletions in amortized $O(\log n)$ time using $O(n)$ space. In the dual space this is equivalent to maintaining the intersection of n half-planes. In order to achieve a total $O(n \log n)$ time, we need to provide a way to traverse recursive visibility regions and guarantee that the number of updates (insertions or deletions of half-planes) in the data structure is $O(n)$. In the rest of this section, we provide a proof for the following lemma.

Algorithm 1 Inverse attraction region.

Input: Simple polygon P, and a point $p \in P$.
Output: Inverse attraction region of p.

```
Compute SPT_r(p) and SPM(p).
for each edge e ∈ SPT_r(p) do
    Compute constraining half-planes of e and corresponding domain subpolygons.
end for
for each region R_i of SPM(p) with base vertex v_i do
    Find all the domain subpolygons that contain v_i, and compute Free_i.
    Intersect R with Free_i, and add the resulting set to the inverse attraction region of p.
end for
return Inverse attraction region of p.
```

▶ **Lemma 16.** *Free spaces of the recursive visibility regions can be computed in a total time of $O(n \log n)$ using $O(n)$ space.*

Proof. Consider a region R_i of $SPM(p)$ with a base vertex v_i. By Lemma 14 and Theorem 8, the set of constraining half-planes that affect the inverse attraction region inside R_i corresponds to the domain subpolygons that contain v_i.

Observe that the vertices of a domain subpolygon appear as one continuous interval along the boundary of P, as there is only one boundary segment of the subpolygon that crosses P. Then, when walking along the boundary of P, each domain subpolygon can be entered and exited at most once. All the domain polygons can be computed in $O(n \log n)$ time by shooting n rays and computing their intersection points with the boundary of P [8].

Let the vertices of P be ordered in the counter-clockwise order. For each domain subpolygon P_e, mark the two endpoints (*e.g.*, vertices v and z in Figure 6) of the boundary edge that crosses P as the first and the last vertices of P_e in accordance to the counter-clockwise order. Then, to obtain the optimal running time, we modify the second for-loop of the Algorithm 1 in the following way. Start at any vertex v_0 of P, find all the domain subpolygons that contain v_0, and initialize the dynamic convex hull data structure of Brodal and Jacob [7] with the points dual to the lines supporting the constraining half-planes of the corresponding domain subpolygons. If v_0 is a base vertex of some region R_0 of $SPM(p)$, then compute the intersection of R_0 and the free space $Free_0$) that we obtain from the dynamic convex hull data structure. Walk along the boundary of P in the counter-clockwise direction, adding to the data structure the dual points to the supporting lines of domain polygons being entered, removing from the data structure the dual points to the supporting lines of domain polygons being exited, and computing the intersection of each region of $SPM(p)$ with the free space obtained from the data structure.

The correctness of the algorithm follows from Lemma 15, and the total running time is $O(n \log n)$. Indeed, there will be $O(n)$ updates to the dynamic convex hull data structure, each requiring $O(\log n)$ amortized time. Intersecting free spaces with regions of $SPM(p)$ will take $O(n \log n)$ time in total, as the complexity of $IAR(p)$ is linear. For the pseudocode of the algorithm please refer to the full version of this paper [12]. ◀

5.1 Lower bound

The proof of the following theorem is based on a reduction from the problem of computing the lower envelope of a set of lines, which has a lower bound of $\Omega(n \log n)$ [18].

▶ **Theorem 17.** *Computing the inverse attraction region of a point in a monotone (or a simple polygon) has a lower bound of $\Omega(n \log n)$.*

Proof. Consider a set of lines L. Let l_b and l_s denote the lines in L with the biggest and smallest slope, respectively. Note that the leftmost (rightmost) edge of the lower envelope of L is part of l_b (l_s).

Without loss of generality assume that the slopes of the lines in L are positive and bounden from above by a small constant ε. We construct a monotone polygon as follows. The right part of the polygon is comprised of an axis aligned rectangle R that contains all the intersection points of the lines in L (Figure 10). Note that R can be computed in linear time. To the left of R, we construct a "zigzag" corridor in the following way. For each line l in L, in an arbitrary order, we add a corridor perpendicular to l which extends above the next arbitrarily chosen line (Figure 11). We then add a corridor with slope 1 going downward until it hits the next line. This process is continued for all lines in L.

Let the point p be the leftmost vertex of the upper chain of the corridor structure. Consider the inverse attraction region of p in the resulting monotone polygon. A point in

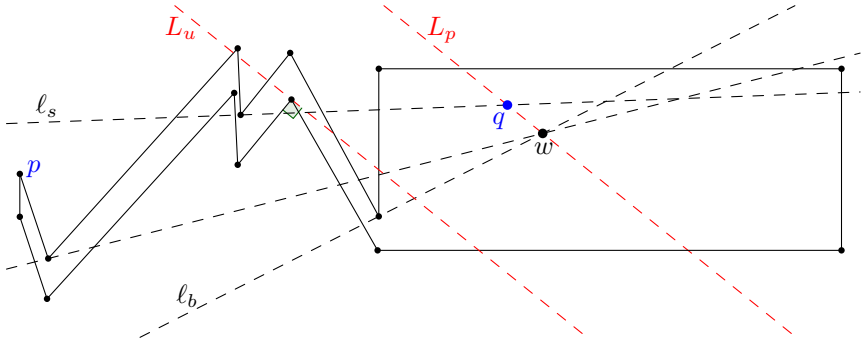


Figure 10 The final monotone polygon constructed for 3 lines.

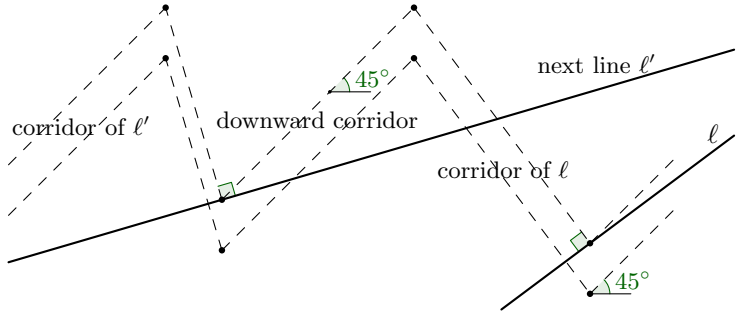


Figure 11 Adding a corridor for a line of L.

R can attract p, only if it is below all lines of L, *i.e.*, only if it is below the lower envelope of L. In addition the point needs to be above the line L_u, where L_u is the rightmost line perpendicular to a lower edge of the corridors with a slope of -1 (refer to Figure 10). In order to have all vertices of the lower envelope in the inverse attraction region, we need to guarantee that L_u is to the left of the leftmost vertex of the lower envelope, w. Let L_p be a line through w with a scope equal to -1. Let q be the intersection of L_p with l_s. We start the first corridor of the zigzag to the left of q. As the lines have similar slopes this guarantees that L_u is to the left of vertices of the lower envelope. Now it is straightforward to compute the lower envelope of L in linear time given the inverse attraction region of p. ◀

We conclude with the main result of this paper.

▶ **Theorem 18.** *The inverse attraction region of a point in a simple polygon can be computed in* $\Theta(n \log n)$ *time.*

References

1 Sang Won Bae, Chan-Su Shin, and Antoine Vigneron. Tight bounds for beacon-based coverage in simple rectilinear polygons. In *Proc. 12th Latin American Symposium on Theoretical Informatics*, 2016.

2 Michael Biro. *Beacon-based routing and guarding.* PhD thesis, Stony Brook University, 2013.

3 Michael Biro, Jie Gao, Justin Iwerks, Irina Kostitsyna, and Joseph S. B. Mitchell. Beacon-based routing and coverage. In *Abstr. 21st Fall Workshop on Computational Geometry*, 2011.

4 Michael Biro, Jie Gao, Justin Iwerks, Irina Kostitsyna, and Joseph S. B. Mitchell. Combinatorics of beacon-based routing and coverage. In *Proc. 25th Canadian Conference on Computational Geometry*, 2013.

5 Michael Biro, Justin Iwerks, Irina Kostitsyna, and Joseph S. B. Mitchell. Beacon-based algorithms for geometric routing. In *Proc. 13th Algorithms and Data Structures Symposium*, 2013.

6 Prosenjit Bose, Pat Morin, Ivan Stojmenović, and Jorge Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wirel. Netw.*, 7(6):609–616, 2001. doi:10.1023/A:1012319418150.

7 Gerth S. Brodal and Riko Jacob. Dynamic planar convex hull. In *Proc. 43rd Annual IEEE Symposium on Foundations of Computer Science*, 2002.

8 Bernard Chazelle and Leonidas J. Guibas. Visibility and intersection problems in plane geometry. *Discrete & Computational Geometry*, 4(6):551–581, 1989.

9 Leonidas J. Guibas, John Hershberger, Daniel Leven, Micha Sharir, and Robert Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2(1–4):209–233, 1987.

10 Brad Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, MobiCom '00, pages 243–254, New York, NY, USA, 2000. ACM. doi:10.1145/345910.345953.

11 Anne-Marie Kermarrec and Guang Tan. Greedy geographic routing in large-scale sensor networks: A minimum network decomposition approach. *IEEE/ACM Transactions on Networking*, 20:864–877, 2010.

12 Irina Kostitsyna, Bahram Kouhestani, Stefan Langerman, and David Rappaport. An optimal algorithm to compute the inverse beacon attraction region. *ArXiv e-prints*, http://arxiv.org/abs/1803.05946, 2018. arXiv:1803.05946.

13 Bahram Kouhestani, David Rappaport, and Kai Salomaa. The length of the beacon attraction trajectory. In *Proc. 27th Canadian Conference on Computational Geometry*, 2015.

14 Bahram Kouhestani, David Rappaport, and Kai Salomaa. On the inverse beacon attraction region of a point. In *Proc. 27th Canadian Conference on Computational Geometry*, 2015.

15 Bahram Kouhestani, David Rappaport, and Kai Salomaa. Routing in a polygonal terrain with the shortest beacon watchtower. *International Journal of Computational Geometry & Applications*, 68:34–47, 2018.

16 Der-Tsai Lee and Franco P. Preparata. Euclidean shortest paths in the presence of rectilinear barriers. *Networks*, 14(3):393–410, 1984.

17 Thomas Shermer. A combinatorial bound for beacon-based routing in orthogonal polygons. In *Proc. 27th Canadian Conference on Computational Geometry*, 2015.

18 Andrew C. Yao. A lower bound to finding convex hulls. *Journal of the ACM*, 28(4):780–787, 1981.

On Optimal Polyline Simplification Using the Hausdorff and Fréchet Distance

Marc van Kreveld[1]
Department of Information and Computing Sciences, Utrecht University
Utrecht, The Netherlands
m.j.vankreveld@uu.nl

Maarten Löffler[2]
Department of Information and Computing Sciences, Utrecht University
Utrecht, The Netherlands
m.loffler@uu.nl

Lionov Wiratma[3]
Department of Information and Computing Sciences, Utrecht University
Utrecht, The Netherlands
Department of Informatics, Parahyangan Catholic University
Bandung, Indonesia
l.wiratma@uu.nl;lionov@unpar.ac.id

Abstract

We revisit the classical polygonal line simplification problem and study it using the Hausdorff distance and Fréchet distance. Interestingly, no previous authors studied line simplification under these measures in its pure form, namely: for a given $\varepsilon > 0$, choose a minimum size subsequence of the vertices of the input such that the Hausdorff or Fréchet distance between the input and output polylines is at most ε.

We analyze how the well-known Douglas-Peucker and Imai-Iri simplification algorithms perform compared to the optimum possible, also in the situation where the algorithms are given a considerably larger error threshold than ε. Furthermore, we show that computing an optimal simplification using the undirected Hausdorff distance is NP-hard. The same holds when using the directed Hausdorff distance from the input to the output polyline, whereas the reverse can be computed in polynomial time. Finally, to compute the optimal simplification from a polygonal line consisting of n vertices under the Fréchet distance, we give an $O(kn^5)$ time algorithm that requires $O(kn^2)$ space, where k is the output complexity of the simplification.

2012 ACM Subject Classification Theory of computation → Computational geometry

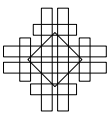
Keywords and phrases polygonal line simplification, Hausdorff distance, Fréchet distance, Imai-Iri, Douglas-Peucker

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.56

Related Version A full version of this paper is available at [23], `http://arxiv.org/abs/1803.03550`.

[1] Supported by The Netherlands Organisation for Scientfic Research on grant no. 612.001.651
[2] Supported by The Netherlands Organisation for Scientfic Research on grant no. 614.001.504
[3] Supported by The Ministry of Research, Technology and Higher Education of Indonesia (No. 138.41/E4.4/2015)

34th International Symposium on Computational Geometry (SoCG 2018).
Editors: Bettina Speckmann and Csaba D. Tóth; Article No. 56; pp. 56:1–56:14



Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Line simplification (a.k.a. polygonal approximation) is one of the oldest and best studied applied topics in computational geometry. It was and still is studied, for example, in the context of computer graphics (after image to vector conversion), in Geographic Information Science, and in shape analysis. Among the well-known algorithms, the ones by Douglas and Peucker [11] and by Imai and Iri [18] hold a special place and are frequently implemented and cited. Both algorithms start with a polygonal line (henceforth *polyline*) as the input, specified by a sequence of points $\langle p_1, \ldots, p_n \rangle$, and compute a subsequence starting with p_1 and ending with p_n, representing a new, simplified polyline. Both algorithms take a constant $\varepsilon > 0$ and guarantee that the output is within ε from the input.

The Douglas-Peucker algorithm [11] is a simple and effective recursive procedure that keeps on adding vertices from the input polyline until the computed polyline lies within a prespecified distance ε. The procedure is a heuristic in several ways: it does not minimize the number of vertices in the output (although it performs well in practice) and it runs in $O(n^2)$ time in the worst case (although in practice it appears more like $O(n \log n)$ time). Hershberger and Snoeyink [17] overcame the worst-case running time bound by providing a worst-case $O(n \log n)$ time algorithm using techniques from computational geometry, in particular a type of dynamic convex hull.

The Imai-Iri algorithm [18] takes a different approach. It computes for every *link* $\overline{p_i p_j}$ with $i < j$ whether the sequence of vertices $\langle p_{i+1}, \ldots, p_{j-1} \rangle$ that lie in between in the input lie within distance ε to the segment $\overline{p_i p_j}$. In this case $\overline{p_i p_j}$ is a valid link that may be used in the output. The graph G that has all vertices $p_1, \ldots, p_n$ as nodes and all valid links as edges can then be constructed, and a minimum link path from p_1 to p_n represents an optimal simplification. Brute-force, this algorithm runs in $O(n^3)$ time, but with the implementation of Chan and Chin [8] or Melkman and O'Rourke [21] it can be done in $O(n^2)$ time.

There are many more results in line simplification. Different error measures can be used [6], self-intersections may be avoided [10], line simplification can be studied in the streaming model [1], it can be studied for 3-dimensional polylines [5], angle constraints may be put on consecutive segments [9], there are versions that do not output a subset of the input points but other well-chosen points [16], it can be incorporated in subdivision simplification [12, 13, 16], and so on and so forth. Some optimization versions are NP-hard [12, 16]. It is beyond the scope of this paper to review the very extensive literature on line simplification.

Among the distance measures for two shapes that are used in computational geometry, the *Hausdorff distance* and the *Fréchet distance* are probably the most well-known. They are both *bottleneck measures*, meaning that the distance is typically determined by a small subset of the input like a single pair of points (and the distances are not aggregated over the whole shapes). The Fréchet distance is considered a better distance measure, but it is considerably more difficult to compute because it requires us to optimize over all parametrizations of the two shapes. The Hausdorff distance between two simple polylines with n and m vertices can be computed in $O((n+m)\log(n+m))$ time [3]. Their Fréchet distance can be computed in $O(nm \log(n+m))$ time [4].

Now, the Imai-Iri algorithm is considered an optimal line simplification algorithm, because it minimizes the number of vertices in the output, given the restriction that the output must be a subsequence of the input. But for what measure? It is not optimal for the Hausdorff distance, because there are simple examples where a simplification with fewer vertices can be given that still have Hausdorff distance at most ε between input and output. This comes from the fact that the algorithm uses the Hausdorff distance between a link $\overline{p_i p_j}$ and the

sub-polyline $\langle p_i, \ldots, p_j \rangle$. This is more local than the Hausdorff distance requires, and is more a Fréchet-type of criterion. But the line simplification produced by the Imai-Iri algorithm is also not optimal for the Fréchet distance. In particular, the input and output do not necessarily lie within Fréchet distance ε, because links are evaluated on their Hausdorff distance only.

The latter issue could easily be remedied: to accept links, we require the Fréchet distance between any link $\overline{p_i p_j}$ and the sub-polyline $\langle p_i, \ldots, p_j \rangle$ to be at most ε [2, 15]. This guarantees that the Fréchet distance between the input and the output is at most ε. However, it does not yield the optimal simplification within Fréchet distance ε. Because of the nature of the Imai-Iri algorithm, *it requires us to match a vertex p_i in the input to the vertex p_i in the output in the parametrizations, if p_i is used in the output.* This restriction on the parametrizations considered limits the simplification in unnecessary ways. Agarwal et al. [2] refer to a simplification that uses the normal (unrestricted) Fréchet distance with error threshold ε as a *weak ε-simplification under the Fréchet distance.*[4] They show that the Imai-Iri algorithm using the Fréchet distance gives a simplification with no more vertices than an optimal weak $(\varepsilon/4)$-simplification under the Fréchet distance, where the latter need not use the input vertices.

The discussion begs the following questions: How much worse do the known algorithms and their variations perform in theory, when compared to the optimal Hausdorff and Fréchet simplifications? What if the optimal Hausdorff and Fréchet simplifications use a smaller value than ε? As mentioned, Agarwal et al. [2] give a partial answer. How efficiently can the optimal Hausdorff simplification and the optimal Fréchet simplification be computed (when using the input vertices)?

Organization and results. In Section 2 we explain the Douglas-Peucker algorithm and its Fréchet variation; the Imai-Iri algorithm has been explained already. We also show with a small example that the optimal Hausdorff simplification has fewer vertices than the Douglas-Peucker output and the Imai-Iri output, and that the same holds true for the optimal Fréchet simplification with respect to the Fréchet variants.

In Section 3 we will analyze the four algorithms and their performance with respect to an optimal Hausdorff simplification or an optimal Fréchet simplification more extensively. In particular, we address the question how many more vertices the four algorithms need, and whether this remains the case when we use a larger value of ε but still compare to the optimization algorithms that use ε.

In Section 4 we consider both the directed and undirected Hausdorff distance to compute the optimal simplification. We show that only the simplification under the directed Hausdorff distance from the output to the input polyline can be computed in polynomial time, while the rest is NP-hard to compute. In Section 5 we show that the problem can be solved in polynomial time for the Fréchet distance.

Table 1 Algorithmic results.

	Douglas-Peucker	Imai-Iri	Optimal
Hausdorff distance	$O(n \log n)$ [17]	$O(n^2)$ [8]	NP-hard *(this paper)*
Fréchet distance	$O(n^2)$ *(easy)*	$O(n^3)$ [15]	$O(kn^5)$ *(this paper)*

[4] Weak refers to the situation that the vertices of the simplification can lie anywhere.

2 Preliminaries

The line simplification problem takes a maximum allowed error ε and a polyline P defined by a sequence of points $\langle p_1, \ldots, p_n \rangle$, and computes a polyline Q defined by $\langle q_1, \ldots, q_k \rangle$ and the error is at most ε. Commonly the sequence of points defining Q is a subsequence of points defining P, and furthermore, $q_1 = p_1$ and $q_k = p_n$. There are many ways to measure the distance or error of a simplification. The most common measure is a distance, denoted by ε, like the Hausdorff distance or the Fréchet distance (we assume these distance measures are known). Note that the Fréchet distance is symmetric, whereas the Hausdorff distance has a symmetric and an asymmmetric version (the distance from the input to the simplification).

The Douglas-Peucker algorithm for polyline simplification is a simple recursive procedure that works as follows. Let the line segment $\overline{p_1 p_n}$ be the first simplification. If all points of P lie within distance ε from this line segment, then we have found our simplification. Otherwise, let p_f be the furthest point from $\overline{p_1 p_n}$, add it to the simplification, and recursively simplify the polylines $\langle p_1, \ldots, p_f \rangle$ and $\langle p_f, \ldots, p_n \rangle$. Then merge their simplifications (remove the duplicate p_f). It is easy to see that the algorithm runs in $O(n^2)$ time, and also that one can expect a much better performance in practice. It is also straightforward to verify that polyline P has Hausdorff distance (symmetric and asymmetric) at most ε to the output. We denote this simplification by $DP_H(P, \varepsilon)$, and will leave out the arguments P and/or ε if they are understood.

We can modify the algorithm to guarantee a Fréchet distance between P and its simplification of at most ε by testing whether the Fréchet distance between P and its simplification is at most ε. If not, we still choose the most distant point p_f to be added to the simplification (other choices are possible). This modification does not change the efficiency of the Douglas-Peucker algorithm asymptotically as the Fréchet distance between a line segment and a polyline can be determined in linear time. We denote this simplification by $DP_F(P, \varepsilon)$.

We have already described the Imai-Iri algorithm in the previous section. We refer to the resulting simplification as $II_H(P, \varepsilon)$. It has a Hausdorff distance (symmetric and asymmetric) of at most ε and never has more vertices than $DP_H(P, \varepsilon)$. Similar to the Douglas-Peucker algorithm, the Imai-Iri algorithm can be modified for the Fréchet distance, leading to a simplification denoted by $II_F(P, \varepsilon)$.

We will denote the optimal simplification using the Hausdorff distance by $OPT_H(P, \varepsilon)$, and the optimal simplification using the Fréchet distance by $OPT_F(P, \varepsilon)$. In the case of Hausdorff distance, we require P to be within ε of its simplification, so we use the directed Hausdorff distance.

The example in Figure 1 shows that $DP_H(P)$ and $II_H(P)$ – which are both equal to P itself – may use more vertices than $OPT_H(P) = \langle p_1, p_5, p_6, p_7 \rangle$. Similarly, the example in Figure 2 shows that DP_F and II_F may use more vertices than OPT_F.

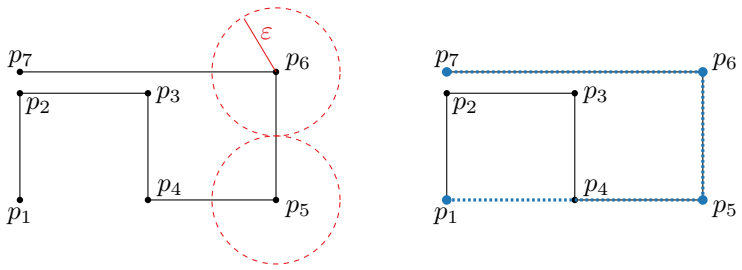


Figure 1 Simplifications II_H (same as input, left) and OPT_H (in blue, right) for an example.

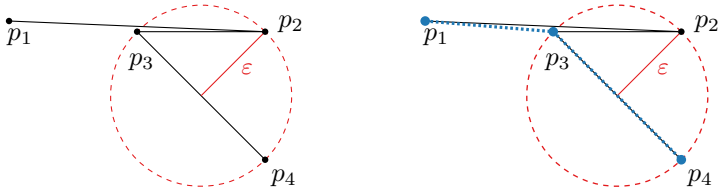


Figure 2 Simplifications II_F (same as input, left) and OPT_F (in blue, right) for an example.

3 Approximation quality of Douglas-Peucker and Imai-Iri simplification

The examples of the previous section not only show that II_H and II_F (and DP_H and DP_F) use more vertices than OPT_H and OPT_F, respectively, they show that this is still the case if we run II with a larger value than ε. To let II_H use as few vertices as OPT_H, we must use 2ε instead of ε when the example is stretched horizontally. For the Fréchet distance, the enlargement factor needed in the example approaches $\sqrt{2}$ if we put p_1 far to the left. In this section we analyze how the approximation enlargement factor relates to the number of vertices in the Douglas-Peucker and Imai-Iri simplifications and the optimal ones. The interest in such results stems from the fact that the Douglas-Peucker and Imai-Iri algorithms are considerably more efficient than the computation of OPT_H and OPT_F.

3.1 Hausdorff distance

To show that II_H (and DP_H by consequence) may use many more vertices than OPT_H, even if we enlarge ε, we give a construction where this occurs. Imagine three regions with diameter ε at the vertices of a sufficiently large equilateral triangle. We construct a polyline P where $p_1, p_5, p_9, \ldots$ are in one region, $p_2, p_4, p_6, \ldots$ are in the second region, and the remaining vertices are in the third region, see Figure 3. Let n be such that p_n is in the third region. An optimal simplification is $\langle p_1, p_i, p_n \rangle$ where i is any even number between 1 and n. Since the only valid links are the ones connecting two consecutive vertices of P, II_H is P itself. If the triangle is large enough with respect to ε, this remains true even if we give the Imai-Iri algorithm a much larger error threshold than ε.

▶ **Theorem 1.** *For any $c > 1$, there exists a polyline P with n vertices and an $\varepsilon > 0$ such that $II_H(P, c\varepsilon)$ has n vertices and $OPT_H(P, \varepsilon)$ has 3 vertices.*

Note that the example applies both to the directed and the undirected Hausdorff distance.

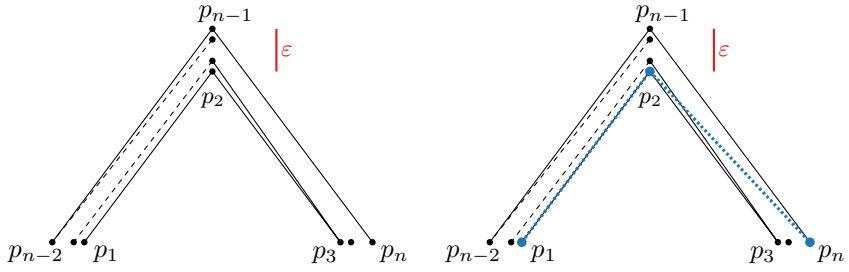


Figure 3 The Douglas-Peucker and Imai-Iri algorithms may not be able to simplify at all, whereas the optimal simplification using the Hausdorff distance has just three vertices (in blue, right).

.

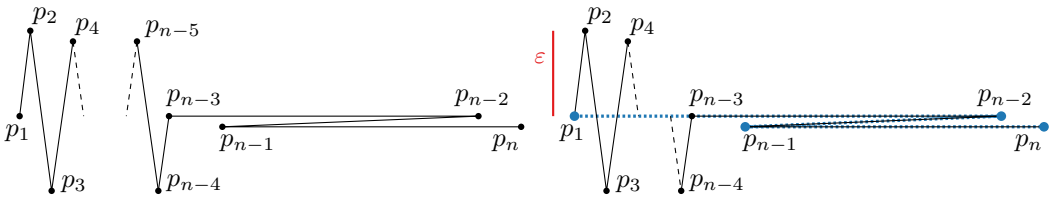


Figure 4 Left: a polyline on which the Fréchet version of the Douglas-Peucker algorithm performs poorly and the output polyline contains n vertices. Right: the optimal simplification contains four vertices (in blue).

3.2 Fréchet distance

Our results are somewhat different for the Fréchet distance; we need to make a distinction between DP_F and II_F.

Douglas-Peucker. We construct an example that shows that DP_F may have many more vertices than OPT_F, even if we enlarge the error threshold. It is illustrated in Figure 4. Vertex p_2 is placed slightly higher than $p_4, p_6, \ldots$ so that it will be added first by the Fréchet version of the Douglas-Peucker algorithm. Eventually all vertices will be chosen. OPT_F has only four vertices. Since the zigzag $p_{n-3}, \ldots, p_n$ can be arbitrarily much larger than the height of the vertical zigzag $p_1, \ldots, p_{n-4}$, the situation remains if we make the error threshold arbitrarily much larger.

▶ **Theorem 2.** *For any $c > 1$, there exists a polyline P with n vertices and an $\varepsilon > 0$ such that $DP_F(P, c\varepsilon)$ has n vertices and $OPT_F(P, \varepsilon)$ has 4 vertices.*

Remark. One could argue that the choice of adding the furthest vertex is not suitable when using the Fréchet distance, because we may not be adding the vertex (or vertices) that are to "blame" for the high Fréchet distance. However, finding the vertex that improves the Fréchet distance most is computationally expensive, defeating the purpose of this simple algorithm. Furthermore, we can observe that also in the Hausdorff version, the Douglas-Peucker algorithm does not choose the vertex that improves the Hausdorff distance most (it may even increase when adding an extra vertex).

Imai-Iri. Finally we compare the Fréchet version of the Imai-Iri algorithm to the optimal Fréchet distance simplification. Our main construction has ten vertices placed in such a way that II_F has all ten vertices, while OPT_F has only eight of them, see Figures 5 and 6.

It is easy to see that under the Fréchet distance, $II_F = OPT_F$ for the previous construction in Figure 4. We give another input polyline P in Figure 6 to show that II_F also does not approximate OPT_F even if II_F is allowed to use ε that is larger by a constant factor.

We can append multiple copies of this construction together with a suitable connection in between. This way we obtain:

▶ **Theorem 3.** *There exist constants $c_1 > 1$, $c_2 > 1$, a polyline P with n vertices, and an $\varepsilon > 0$ such that $|II_F(P, c_1\varepsilon)| > c_2|OPT_F(P, \varepsilon)|$.*

By the aforementioned result of Agarwal et al. [2], we know that the theorem is not true for $c_1 \geq 4$.

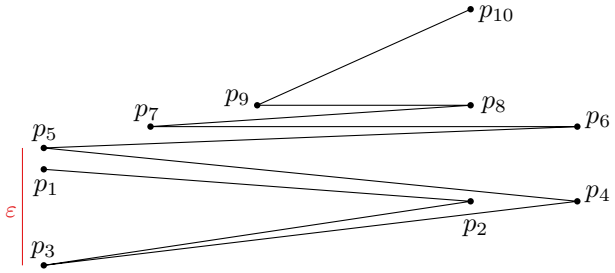


Figure 5 The Imai-Iri simplification will have all vertices because the only valid links with a Fréchet distance at most ε are the ones connecting two consecutive vertices in the polyline.

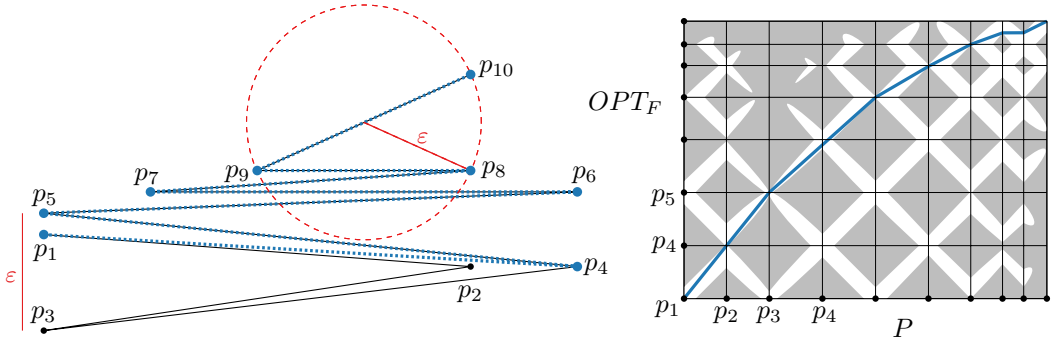


Figure 6 The optimal simplification can skip p_2 and p_3; in the parametrizations witnessing the Fréchet distance, OPT_F "stays two vertices behind" on the input until the end. Right, the free space diagram of P and OPT_F.

4 Algorithmic complexity of the Hausdorff distance

The results in the previous section show that both the Douglas-Peucker and the Imai-Iri algorithm do not produce an optimal polyline that minimizes the Hausdorff or Fréchet distance, or even approximate them within any constant factor. Naturally, this leads us to the following question: Is it possible to compute the optimal Hausdorff or Fréchet simplification in polynomial time?

In this section, we present a construction which proves that under the Hausdorff distance, computing the optimal simplified polyline is NP-hard.

4.1 Undirected Hausdorff distance

We first consider the undirected (or bidirectional) Hausdorff distance; that is, we require both the maximum distance from the initial polyline P to the simplified polyline Q and the maximum distance from Q to P to be at most ε.

▶ **Theorem 4.** *Given a polyline $P = \langle p_1, p_2, \ldots, p_n \rangle$ and a value ε, the problem of computing a minimum length subsequence Q of P such that the undirected Hausdorff distance between P and Q is at most ε is NP-hard.*

We prove the theorem with a reduction from Hamiltonian cycle in segment intersection graphs. It is well-known that Hamiltonian cycle is NP-complete in planar graphs [14], and by Chalopin and Gonçalves' proof [7] of Scheinerman's conjecture [22] that the planar graphs

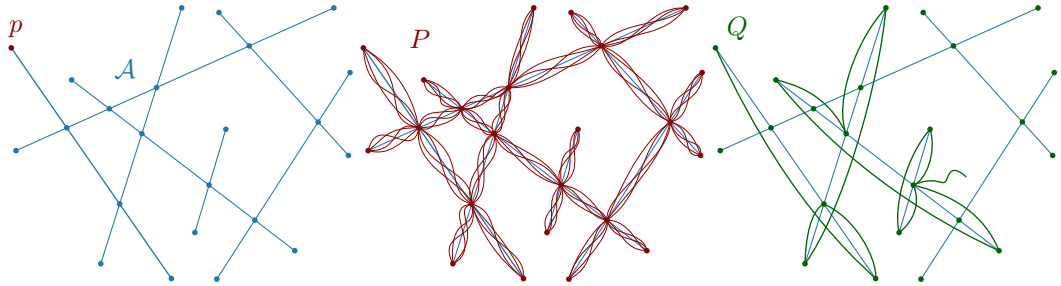


Figure 7 The construction: $\mathcal{A}$ is the arrangement of a set of segments S. We build an input path P that "paints" over S completely, and we are looking for an output path Q that corresponds to a Hamiltonian cycle. In this case, there is no Hamiltonian cycle, and the path gets stuck.

are included in the segment intersections graphs it follows that Hamiltonian cycle in segment intersections graphs is NP-complete.

Let S be a set of n line segments in the plane, and assume all intersections are proper (if not, extend the segments slightly). Let G be its intersection graph (i.e. G has a vertex for every segment in S, and two vertices in G are connected by an edge when their corresponding segments intersect). We assume that G is connected; otherwise, clearly there is no Hamiltonian cycle in G.

We first construct an initial polyline P as follows. (Figure 7 illustrates the construction.) Let $\mathcal{A}$ be the arrangement of S, let p be some endpoint of a segment in S, and let π be any path on $\mathcal{A}$ that starts and finishes at p and visits all vertices and edges of $\mathcal{A}$ (clearly, π may reuse vertices and edges). Then P is simply $3n+1$ copies of π appended to each other. Consequently, the order of vertices in Q now must follow the order of these copies. We now set ε to a sufficiently small value.

Now, an output polyline Q with Hausdorff distance at most ε to P must also visit all vertices and edges of $\mathcal{A}$, and stay close to $\mathcal{A}$. If ε is sufficiently small, there will be no benefit for Q to ever leave $\mathcal{A}$.

▶ **Lemma 5.** *A solution Q of length $3n+1$ exists if and only if G admits a Hamiltonian cycle.*

Proof. Clearly, any simplification Q will need to visit the $2n$ endpoints of the segments in S, and since it starts and ends at the same point p, will need to have length at least $2n+1$. Furthermore, Q will need to have at least two internal vertices on every segment $s \in S$: once to enter the segment and once to leave it (note that we cannot enter or leave a segment at an endpoint since all intersections are proper intersections). This means the minimum number of vertices possible for Q is $3n+1$.

Now, if G admits a Hamiltonian cycle, it is easy to construct a simplification with $3n+1$ vertices as follows. We start at p and collect the other endpoint of the segment s_1 of which p is an endpoint. Then we follow the Hamiltonian cycle to segment s_2; by definition $s_1 s_2$ is an edge in G so their corresponding segments intersect, and we use the intersection point to leave s_1 and enter s_2. We proceed in this fashion until we reach s_n, which intersects s_1, and finally return to p.

On the other hand, any solution with $3n+1$ vertices must necessarily be of this form and therefore imply a Hamiltonian cycle: in order to have only 3 vertices per segment the vertex at which we leave s_1 must coincide with the vertex at which we enter some other segment, which we call s_2, and we must continue until we visited all segments and return to p. ◀

4.2 Directed Hausdorff distance: $P \to Q$

We now shift our attention to the directed Hausdorff distance from P to Q: we require the maximum distance from P to Q to be at most ε, but Q may have a larger distance to P. The previous reduction does not seem to work because there is always a Hamiltonian Cycle of length $2n$ for this measure. Therefore, we prove the NP-hardness differently.

The idea is to reduce from Covering Points By Lines, which is known to be both NP-hard [20] and APX-hard [19]: given a set S of points in $\mathbb{R}^2$, find the minimum number of lines needed to cover the points. The complete proof is explained in full detail in [23]; here we give the main part of the construction.

Let $S = \{s_1, \ldots, s_n\}$ be an instance of the Covering Points By Lines problem. We fix ε based on S and present the construction of a polyline connecting a sequence of $m = \text{poly}(n)$ points: $P = \langle p_1, p_2, ..., p_m \rangle$ such that for every $1 \leq i \leq n$, we have $s_i = p_j$ for some $1 \leq j \leq m$. The idea is to force the simplification Q to cover all points in P except those in S, such that in order for the final simplification to cover all points, we only need to collect the points in S using as few line segments as possible. To this end, we will place a number of *forced points* $F \subset P$, where a point f is *forced* whenever its distance to any line through any pair of points in P is larger than ε. Since Q must be defined by a subset of points in P, we will never cover f unless we choose f to be a vertex of Q. Figure 8 shows this idea. On the other hand, we need to place points that allow us to freely draw every line through two or more points in S. We create two point sets L and R to the left and right of S, such that for every line through two of more points in S, there are a point in L and a point in R on that line. Finally, we need to build additional scaffolding around the construction to connect and cover the points in L and R. Figure 9 shows the idea.

The construction has three parts with different purposes:

1. a sub-polyline that contains S;
2. a sub-polyline that contains L and R; and
3. two disconnected sub-polylines which share the same purpose: to guarantee that all vertices in the previous sub-polyline are themselves covered by Q.

First, we assume that every point in S has a unique x-coordinate; if this is not the case, we rotate S until it is.[5] We also assume that every line through at least two points of S has a slope between -1 and $+1$; if this is not the case, we vertically scale S until it is. Now, we fix ε to be smaller than half the minimum difference between any two x-coordinates of points in S, and smaller than the distance from any line through two points in S to any other point in S not on the line.

We place $n + 1$ forced points $f_1, f_2, ..., f_n, f_{n+1}$ such that the x-coordinate of f_i lies between the x-coordinates of s_{i-1} and s_i and the points lie alternatingly above and below S; we place them such that the distance of the line segment $\overline{f_i f_{i+1}}$ to s_i is $\frac{3}{2}\varepsilon$ and the distance of $\overline{f_i f_{i+1}}$ to s_{i-1} is larger than ε. Next, we place two auxiliary points t_i^+ and t_i^- on $\overline{f_i f_{i+1}}$ such that the distance of each point to s_i is 2ε; refer to Figure 8. Then let $\tau_1 = \langle f_1, t_1^+, s_1, t_1^-, f_2, t_2^-, s_2, t_2^+, f_3, \ldots, f_{n+1} \rangle$ be a polyline connecting all points in the construction; τ_1 will be part of the input segment P.

The idea here is that all forced points must appear on Q, and if only the forced points appear on Q, everything in the construction will be covered *except* the points in S (and some arbitrarily short stubs of edges connecting them to the auxiliary points). Of course, we could

[5] Note that, by nature of the Covering Points By Lines problem, we cannot assume S is in general position; however, a rotation for which all x-coordinates are unique always exists.

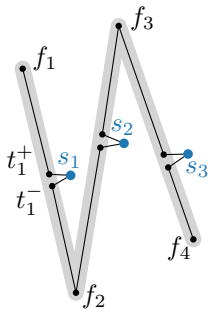


Figure 8 Example of τ_1 where $n = 3$. For a given ε, the (simplified) polyline f_1, f_2, f_3, f_4 covers the gray area but not the blue points s_1, s_2, s_3.

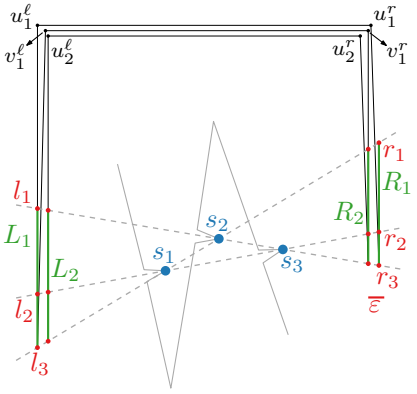


Figure 9 Construction to allow the lines that can be used to cover the points of S. To ensure the order of vertices in Q, we create copies of L and R. Then, Q can use them alternatingly.

choose to include more points in τ_1 in Q to collect some points of S already. However, this would cost an additional three vertices per collected point (note that using fewer than three, we would miss an auxiliary point instead), and in the remainder of the construction we will make sure that it is cheaper to collect the points in S separately later.

The second part of the construction serves to allow shortcuts that have the role of the lines in the Covering Points By Lines problem. Since we only need the $O(n^2)$ lines that cover at least two points, this part of the construction has $O(n^2)$ vertices in P. An indication of this construction is given in Figure 9; further details are in [23].

▶ **Theorem 6.** *Given a polyline $P = \langle p_1, p_2, \ldots, p_n \rangle$ and a value ε, the problem of computing a minimum length subsequence Q of P such that the directed Hausdorff distance from P to Q is at most ε is NP-hard.*

4.3 Directed Hausdorff distance: $Q \to P$

Finally, we finish this section with a note on the reverse problem: we want to only bound the directed Hausdorff distance from Q to P (we want the output segment to stay close to the input segment, but we do not need to be close to all parts of the input). This problem seems more esoteric but we include it for completeness. In this case, a polynomial time algorithm (reminiscent of Imai-Iri) optimally solves the problem.

▶ **Theorem 7.** *Given a polyline $P = \langle p_1, p_2, \ldots, p_n \rangle$ and a value ε, the problem of computing a minimum length subsequence Q of P such that the directed Hausdorff distance from Q to P is at most ε can be solved in polynomial time.*

Proof. We compute the region with distance ε from P explicitly. For every link we compute if it lies within that region, and if so, add it as an edge to a graph. Then we find a minimum link path in this graph. For a possibly self-intersecting polyline as the input a simple algorithm takes $O(n^4)$ time (faster is possible). ◀

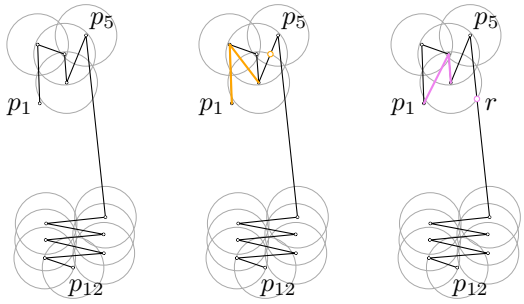


Figure 10 An example where the farthest-reaching simplification up to p_4 using 2 links is not part of any solution that uses p_4. Left: the input curve P in black, with circles of radius ε around all vertices in light gray. Middle: A 2-link simplification of $\langle p_1, p_2, p_3, p_4\rangle$ that reaches up to a point on $\overline{p_4p_5}$ (in yellow) which can be extended to a 4-link simplification of P. Right: A 2-link simplification of $\langle p_1, p_2, p_3, p_4\rangle$ that reaches point r on $\overline{p_5p_6}$ (in pink) which does not allow simplification.

5 Algorithmic complexity of the Fréchet distance

In this section, we show that for a given polyline $P = \langle p_1, p_2, ..., p_n\rangle$ and an error ε, the optimal simplification $Q = OPT_F(P, \varepsilon)$ can be computed in polynomial time using a dynamic programming approach.

5.1 Observations

Note that a link $\overline{p_ip_j}$ in Q is not necessarily within Fréchet distance ε to the sub-polyline $\langle p_i, p_{i+1}, ..., p_j\rangle$ (for example, $\overline{p_1p_3}$ in Figure 2). Furthermore, a (sequence of) link(s) in Q could be mapped to an arbitrary subcurve of P, not necessarily starting or ending at a vertex of P. For example, in Figure 6, the sub-polyline $\langle p_1, p_4, p_5, p_6\rangle$ has Fréchet distance ε to a sub-polyline of P that starts at p_1 but ends somewhere between p_4 and p_5. At this point, one might imagine a dynamic programming algorithm which stores, for each vertex p_i and value k, the point $p(i, k)$ on P which is the farthest along P such that there exists a simplification of the part of P up to p_i using k links that has Fréchet distance at most ε to the part of P up to $p(i, k)$. However, the following lemma shows that even this does not yield optimality; its proof is the example in Figure 10.

▶ **Lemma 8.** *There exists a polyline $P = \langle p_1, \ldots, p_{12}\rangle$ and an optimal ε-Fréchet-simplification that has to use p_4, $Q = \langle p_1, p_2, p_4, p_5, p_{12}\rangle$ using 4 links, with the following properties:*

- *There exists a partial simplification $R = \langle p_1, p_3, p_4\rangle$ of $\langle p_1, \ldots, p_4\rangle$ and a point r on $\overline{p_5p_6}$ such that the Fréchet distance between R and the subcurve of P up to r is $\leq \varepsilon$, but*
- *there exists no partial simplification S of $\langle p_4, \ldots, p_{12}\rangle$ that is within Fréchet distance ε to the subcurve of P starting at r that uses fewer than 7 links.*

5.2 A dynamic programming algorithm

Lemma 8 shows that storing a single data point for each vertex and value of k is not sufficient to ensure that we find an optimal solution. Instead, we argue that if we maintain the set of *all* points at P that can be "reached" by a simplification up to each vertex, then we can make dynamic programming work. We now make this precise and argue that the complexity of these sets of reachable points is never worse than linear.

First, we define π, a parameterization of P as a continuous mapping: $\pi : [0,1] \to \mathbb{R}^2$ where $\pi(0) = p_1$ and $\pi(1) = p_n$. We also write $P[s,t]$ for $0 \leq s \leq t \leq 1$ to be the subcurve of P starting at $\pi(s)$ and ending at $\pi(t)$, also writing $P[t] = P[0,t]$ for short.

We say that a point $\pi(t)$ can be *reached* by a (k,i)-simplification for $0 \leq k < i \leq n$ if there exists a simplification of $\langle p_1, \ldots, p_i \rangle$ using k links which has Fréchet distance at most ε to $P[t]$. We let $\rho(k,i,t) = \texttt{true}$ in this case, and `false` otherwise. With slight abuse of notation we also say that t itself is reachable, and that an interval I is reachable if all $t \in I$ are reachable (by a (k,i)-simplification).

▶ **Obervation 9.** *A point $\pi(t)$ can be reached by a (k,i)-simplification if and only if there exist a $0 < h < i$ and a $0 \leq s \leq t$ such that $\pi(s)$ can be reached by a $(k-1,h)$-simplification and the segment $\overline{p_h p_i}$ has Fréchet distance at most ε to $P[s,t]$.*

Proof. Follows directly from the definition of the Fréchet distance. ◀

Observation 9 immediately suggests a dynamic programming algorithm: for every k and i we store a subdivision of $[0,1]$ into intervals where ρ is true and intervals where ρ is false, and we calculate the subdivisions for increasing values of k. We simply iterate over all possible values of h, calculate which intervals can be reached using a simplification via h, and then take the union over all those intervals. For this, the only unclear part is how to calculate these intervals.

We argue that, for any given k and i, there are at most $n-1$ reachable intervals on $[0,1]$, each contained in an edge of P. Indeed, every (k,i)-reachable point $\pi(t)$ must have distance at most ε to p_i, and since the edge e of P that $\pi(t)$ lies on intersects the disk of radius ε centered at p_i in a line segment, every point on this segment is also (k,i)-reachable. We denote the farthest point on e which is (k,i)-reachable by $\hat{t}$.

Furthermore, we argue that for each edge of P, we only need to take the farthest reachable point into account during our dynamic programming algorithm.

▶ **Lemma 10.** *If k, h, i, s, and t exist such that $\rho(k-1,h,s) = \rho(k,i,t) = \texttt{true}$, and $\overline{p_h p_i}$ has Fréchet distance $\leq \varepsilon$ to $P[s,t]$, then $\overline{p_h p_i}$ also has Fréchet distance $\leq \varepsilon$ to $P[\hat{s},\hat{t}]$.*

Proof. By the above argument, $P[s,\hat{s}]$ is a line segment that lies completely within distance ε from p_h, and $P[t,\hat{t}]$ is a line segment that lies completely within distance ε from p_i.

We are given that the Fréchet distance between $\overline{p_h p_i}$ and $P[s,t]$ is at most ε; this means a mapping $f : [s,t] \to \overline{p_h p_i}$ exists such that $|\pi(x) - f(x)| \leq \varepsilon$. Let $q = f(s')$. Then $|p_h - \pi(\hat{s})| \leq \varepsilon$ and $|q - \pi(\hat{s})| \leq \varepsilon$, so the line segment $\overline{p_h q}$ lies fully within distance ε from $\hat{s}$.

Therefore, we can define a new ε-Fréchet mapping between $P[\hat{s},\hat{t}]$ and $\overline{p_h p_i}$ which maps $\hat{s}$ to the segment $\overline{p_h q}$, the curve $P[\hat{s},t]$ to the segment $\overline{q p_i}$ (following the mapping given by f), and the segment $\overline{\pi(t)\pi(\hat{t})}$ to the point p_i. ◀

Now, we can compute the optimal simplification by maintaining a $k \times n \times n$ table storing $\rho(k,i,\hat{t})$, and calculate each value by looking up n^2 values for the previous value of k, and testing in linear time for each combination whether the Fréchet distance between the new link and $P[\hat{s},\hat{t}]$ is within ε or not.

▶ **Theorem 11.** *Given a polyline $P = \langle p_1, \ldots, p_n \rangle$ and a value ε, we can compute the optimal polyline simplification of P that has Fréchet distance at most ε to P in $O(kn^5)$ time and $O(kn^2)$ space, where k is the output complexity of the optimal simplification.*

6 Conclusions

In this paper, we analyzed well-known polygonal line simplification algorithms, the Douglas-Peucker and the Imai-Iri algorithm, under both the Hausdorff and the Fréchet distance. Both algorithms are not optimal when considering these measures. We studied the relation between the number of vertices in the resulting simplified polyline from both algorithms and the enlargement factor needed to approximate the optimal solution. For the Hausdorff distance, we presented a polyline where the optimal simplification uses only a constant number of vertices while the solution from both algorithms is the same as the input polyline, even if we enlarge ε by any constant factor. We obtain the same result for the Douglas-Peucker algorithm under the Fréchet distance. For the Imai-Iri algorithm, such a result does not exist but we have shown that we will need a constant factor more vertices if we enlarge the error threshold by some small constant, for certain polylines.

Next, we investigated the algorithmic problem of computing the optimal simplification using the Hausdorff and the Fréchet distance. For the directed and undirected Hausdorff distance, we gave NP hardness proofs. Interestingly, the optimal simplification in the other direction (from output to input) is solvable in polynomial time. Finally, we showed how to compute the optimal simplification under the Fréchet distance in polynomial time. Our algorithm is based on the dynamic programming method and runs in $O(kn^5)$ time and requires $O(kn^2)$ space.

A number of challenging open problems remain. First, we would like to show NP-hardness of computing an optimal simplification using the Hausdorff distance when the simplification may not have self-intersections. Second, we are interested in the computational status of the optimal simplification under the Hausdorff distance and the Fréchet distance when the simplification need not use the vertices of the input. Third, it is possible that the efficiency of our algorithm for computing an optimal simplification with Fréchet distance at most ε can be improved. Fourth, we may consider optimal polyline simplifications using the weak Fréchet distance.

References

1 Mohammad Ali Abam, Mark de Berg, Peter Hachenberger, and Alireza Zarei:. Streaming algorithms for line simplification. *Discrete & Computational Geometry*, 43(3):497–515, 2010.

2 Pankaj K. Agarwal, Sariel Har-Peled, Nabil H. Mustafa, and Yusu Wang. Near-linear time approximation algorithms for curve simplification. *Algorithmica*, 42(3):203–219, 2005.

3 Helmut Alt, Bernd Behrends, and Johannes Blömer. Approximate matching of polygonal shapes. *Annals of Mathematics and Artificial Intelligence*, 13(3):251–265, Sep 1995.

4 Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 5(1-2):75–91, 1995.

5 Gill Barequet, Danny Z. Chen, Ovidiu Daescu, Michael T. Goodrich, and Jack Snoeyink. Efficiently approximating polygonal paths in three and higher dimensions. *Algorithmica*, 33(2):150–167, 2002.

6 Lilian Buzer. Optimal simplification of polygonal chain for rendering. In *Proceedings 23rd Annual ACM Symposium on Computational Geometry*, SCG '07, pages 168–174, 2007.

7 Jérémie Chalopin and Daniel Gonçalves. Every planar graph is the intersection graph of segments in the plane: Extended abstract. In *Proceedings 41st Annual ACM Symposium on Theory of Computing*, STOC '09, pages 631–638, 2009.

8 W.S. Chan and F. Chin. Approximation of polygonal curves with minimum number of line segments or minimum error. *International Journal of Computational Geometry & Applications*, 06(01):59–77, 1996.

9 Danny Z. Chen, Ovidiu Daescu, John Hershberger, Peter M. Kogge, Ningfang Mi, and Jack Snoeyink. Polygonal path simplification with angle constraints. *Computational Geometry*, 32(3):173–187, 2005.

10 Mark de Berg, Marc van Kreveld, and Stefan Schirra. Topologically correct subdivision simplification using the bandwidth criterion. *Cartography and Geographic Information Systems*, 25(4):243–257, 1998.

11 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*, 10(2):112–122, 1973.

12 Regina Estkowski and Joseph S. B. Mitchell. Simplifying a polygonal subdivision while keeping it simple. In *Proceedings 17th Annual ACM Symposium on Computational Geometry*, SCG '01, pages 40–49, 2001.

13 Stefan Funke, Thomas Mendel, Alexander Miller, Sabine Storandt, and Maria Wiebe. Map simplification with topology constraints: Exactly and in practice. In *Proc. 19th Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 185–196, 2017.

14 M.R. Garey, D.S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976.

15 Michael Godau. A natural metric for curves - computing the distance for polygonal chains and approximation algorithms. In *Proceedings 8th Annual Symposium on Theoretical Aspects of Computer Science*, STACS 91, pages 127–136. Springer-Verlag, 1991.

16 Leonidas J. Guibas, John E. Hershberger, Joseph S.B. Mitchell, and Jack Scott Snoeyink. Approximating polygons and subdivisions with minimum-link paths. *International Journal of Computational Geometry & Applications*, 03(04):383–415, 1993.

17 John Hershberger and Jack Snoeyink. An $O(n \log n)$ implementation of the Douglas-Peucker algorithm for line simplification. In *Proceedings 10th Annual ACM Symposium on Computational Geometry*, SCG '94, pages 383–384, 1994.

18 Hiroshi Imai and Masao Iri. Polygonal approximations of a curve - formulations and algorithms. In Godfried T. Toussaint, editor, *Computational Morphology: A Computational Geometric Approach to the Analysis of Form*. North-Holland, Amsterdam, 1988.

19 V. S. Anil Kumar, Sunil Arya, and H. Ramesh. Hardness of set cover with intersection 1. In *Automata, Languages and Programming: 27th International Colloquium, ICALP 2000*, pages 624–635. Springer, Berlin, Heidelberg, 2000.

20 Nimrod Megiddo and Arie Tamir. On the complexity of locating linear facilities in the plane. *Operations Research Letters*, 1(5):194–197, 1982.

21 Avraham Melkman and Joseph O'Rourke. On polygonal chain approximation. In Godfried T. Toussaint, editor, *Computational Morphology: A Computational Geometric Approach to the Analysis of Form*, pages 87–95. North-Holland, Amsterdam, 1988.

22 E. R. Scheinerman. *Intersection Classes and Multiple Intersection Parameters of Graphs*. PhD thesis, Princeton University, 1984.

23 Marc van Kreveld, Maarten Löffler, and Lionov Wiratma. On optimal polyline simplification using the hausdorff and fréchet distance. URL: `https://arxiv.org/abs/1803.03550`.

Graph-Based Time–Space Trade-Offs for Approximate Near Neighbors

Thijs Laarhoven
Eindhoven University of Technology
Eindhoven, The Netherlands
mail@thijs.com
https://orcid.org/0000-0002-2369-9067

Abstract

We take a first step towards a rigorous asymptotic analysis of graph-based methods for finding (approximate) nearest neighbors in high-dimensional spaces, by analyzing the complexity of randomized greedy walks on the approximate nearest neighbor graph. For random data sets of size $n = 2^{o(d)}$ on the d-dimensional Euclidean unit sphere, using near neighbor graphs we can provably solve the approximate nearest neighbor problem with approximation factor $c > 1$ in query time $n^{\rho_q + o(1)}$ and space $n^{1+\rho_s+o(1)}$, for arbitrary $\rho_q, \rho_s \geq 0$ satisfying

$$(2c^2 - 1)\rho_q + 2c^2(c^2 - 1)\sqrt{\rho_s(1 - \rho_s)} \geq c^4. \tag{1}$$

Graph-based near neighbor searching is especially competitive with hash-based methods for small c and near-linear memory, and in this regime the asymptotic scaling of a greedy graph-based search matches optimal hash-based trade-offs of Andoni–Laarhoven–Razenshteyn–Waingarten [5]. We further study how the trade-offs scale when the data set is of size $n = 2^{\Theta(d)}$, and analyze asymptotic complexities when applying these results to lattice sieving.

2012 ACM Subject Classification Theory of computation → Nearest neighbor algorithms

Keywords and phrases approximate nearest neighbor problem, near neighbor graphs, locality-sensitive hashing, locality-sensitive filters, similarity search

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.57

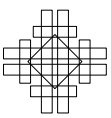
Related Version A full version, containing proofs of all claims and discussing the application of these results to lattice sieving, is available online: `https://arxiv.org/abs/1712.03158`.

Funding This work was supported by ERC consolidator grant 617951.

Acknowledgements The author thanks Marek Eliáš, Jesper Nederlof, and Jorn van der Pol for enlightening discussions and comments on the proofs in the appendices (see the full version).

1 Introduction

Nearest neighbor searching. A key computational problem in various areas of research, such as machine learning, pattern recognition, data compression, and decoding [15, 26, 27, 38, 33, 45], is the *nearest neighbor problem*: given a d-dimensional data set $\mathcal{D} \subset \mathbb{R}^d$ of cardinality n, design a data structure and preprocess $\mathcal{D}$ in an efficient way such that, when later given a query vector $\boldsymbol{q} \in \mathbb{R}^d$, we can quickly find the nearest point to $\boldsymbol{q}$ in $\mathcal{D}$ (e.g. under the Euclidean metric). Since the exact (worst-case) version of this problem suffers from the "curse of dimensionality" [30], a common relaxation of this problem is the *approximate nearest neighbor (ANN) problem*: given that the exact nearest neighbor in the data set $\mathcal{D}$ lies at distance at most r from $\boldsymbol{q}$, design an efficient algorithm that finds an element $\boldsymbol{p} \in \mathcal{D}$




at distance at most $c \cdot r$ from $\boldsymbol{q}$, for a given approximation factor $c > 1$. We will refer to this problem as (c, r)-ANN. Since a naive linear search trivially leads to an $O(d \cdot n)$ time algorithm for solving both the exact and approximate near neighbor problems, the goal is to design a data structure for which queries can be accurately answered in time $n^{\rho+o(1)}$ with $\rho < 1$. Here we will only consider scenarios where d scales with n – for fixed d, it is well-known that one can achieve arbitrarily small query exponents $\rho = o(1)$ [8]. We further assume w.l.o.g. that $n \geq 2^{O(d/\log d)}$ – in case d is larger, we can first perform a random projection to reduce the effective dimensionality of the data set, while maintaining inter-point distances [31].

Partitioning the space. A celebrated technique for efficiently solving ANN in high dimensions is *locality-sensitive hashing (LSH)* [30, 23, 2, 4]. Using hash functions with the property that nearby vectors are more likely to be mapped to the same hash value, one builds several hash tables with buckets containing vectors with the same hash values. Queries $\boldsymbol{q}$ are then processed by computing $h(\boldsymbol{q})$, looking up vectors $\boldsymbol{p} \in \mathcal{D}$ with the same hash value $h(\boldsymbol{p}) = h(\boldsymbol{q})$, and considering these vectors as candidate near neighbors, for each of the precomputed hash tables. Although many hash tables are required to obtain a good recall rate, doing these look-ups in the hash tables is often considerably faster than a linear search through the list.

Whereas LSH requires each point to be mapped to exactly one hash bucket in each hash table, the recent *locality-sensitive filtering (LSF)* [11, 5, 20] relaxes this condition: for each hash filter and each point in the data set, we independently decide whether we add this point to this filter bucket or not. This may mean that some points are added to more filters than others, and filters do not necessarily form a partitioning of the space. Queries are answered by considering filters matching the query, and going through all vectors in these buckets.

For the special case of ANN where the data set lies on the sphere, many efficient partition-based methods are known based on using random hyperplanes [18], regular polytopes [46, 3, 32, 35], and spherical caps [2, 4, 11, 34, 5]. For random data sets of size $n = 2^{o(d)}$, both cross-polytope LSH [3] and spherical LSF [11] achieve the optimal asymptotic scaling of the query complexity for hash-based methods. Spherical LSF further offers optimal time–space trade-offs between the query and update/space complexities [5], while cross-polytope LSH seems to perform better in practice [3, 44, 9]. No optimality results are known for data sets of size $n = 2^{\Theta(d)}$, but spherical LSF outperforms cross-polytope LSH in some regimes [11].

Nearest neighbor graphs. A different approach to the near neighbor problem involves constructing *nearest neighbor graphs*, where vertices correspond to points $\boldsymbol{p} \in \mathcal{D}$ of the data set, and an edge between two vertices indicates that these points are approximate near neighbors [17, 21, 28, 39, 42, 19, 22, 25, 29, 47]. Given a query $\boldsymbol{q}$, one starts at an arbitrary node $\boldsymbol{p} \in \mathcal{D}$, and repeatedly attempts to find vertices $\boldsymbol{p}'$, connected to $\boldsymbol{p}$ in the graph through an edge, which are closer to $\boldsymbol{q}$ than the current candidate near neighbor $\boldsymbol{p}$. When this process terminates, the resulting vector is either a local or a global minimum, i.e. a false positive or the true nearest neighbor to $\boldsymbol{q}$. If the graph is sufficiently well-connected, we may hope to solve the (A)NN problem with high probability in one iteration – otherwise we might start over with a new random $\boldsymbol{p} \in \mathcal{D}$, and hope for success in a reasonable number of attempts.

Compared to LSH and LSF, which often require storing quite some auxiliary data describing the hash tables/filters, and for which the cost of computing hashes or finding appropriate filters is commonly non-negligible, the nearest neighbor graph approach has much less overhead: for each vector $\boldsymbol{p} \in \mathcal{D}$, one only has to store its nearest neighbors in a list, and look-ups require no additional computations besides comparisons between the query $\boldsymbol{q}$ and data points $\boldsymbol{p} \in \mathcal{D}$ in these lists to find nearer neighbors. In practice, graph-based approaches may therefore be more efficient than hash-based methods even if asymptotic analyses suggest otherwise, purely due to the low overhead of graph-based methods.

Besides the standard nearest neighbor graph approach of connecting each vertex to its nearest neighbors [25, 24], various heuristic graph-based methods are further known based on adding connections in the graph between points which are not necessaily near neighbors [43, 37, 16]. This might for instance involve including several long-range, deep links in the graph between distant points, to guarantee that every pair of nodes is connected through a short path [37]. Using hierarchical graphs [36, 16] where vertices are partitioned in several layers further seems to aid the performance of graph-based methods, although again these improvements appear to be purely heuristic.

Comparison of different methods. To find out which method is objectively the best, an obvious approach would be to implement these methods, test them against realistic data sets, and compare their concrete performance. Various libraries with implementations of near neighbor methods are openly available online [14, 44, 40, 24, 16], and recently Aumüller–Bernhardsson–Faithfull presented a thorough comparison of different methods for nearest neighbor searching on various commonly used data sets and distance metrics [13, 9]. Their final conclusions include that the LSH-based `FALCONN` and the graph-based `NMSLib` and `KGraph` currently seem to be the most competitive for their data sets.

Practical comparisons clearly have various drawbacks, as the practical performance often depends on many additional variables, such as specific properties of the data set and the level of optimization of the implementation – better results on certain data sets may not always translate well to other data sets with further optimized implementations. Moreover, some methods may scale better as the dimensionality and size of the data set increases, and it is hard to accurately predict asymptotic behavior from practical experiments.

A second approach for comparing different methods would be to look at their theoretical, proven asymptotic performance as the size of the data set n and the dimension d tend to infinity. Tight bounds on the performance would allow us to extrapolate to arbitrary data sets, but for many algorithms obtaining such tight asymptotic bounds appears challenging. Although for partition-based methods, by now the asymptotic performance seems to be reasonably well understood, graph-based algorithms are mostly based on unproven heuristics, and are not as well understood theoretically. This is particularly disconcerting due to the efficiency of certain graph-based approaches, and so a natural question is: how well do graph-based approaches hold up against partition-based methods in high dimensions? Is it just the low overhead of graph-based methods that allows them to compete with hash-based approaches? Or will these graph-based methods remain competitive even for huge data sets?

Heuristic methods made theoretical. We further remark that in the past, theoretical analyses of heuristic near neighbor methods have not only contributed to a better understanding of these methods, but also to make these methods more practical. Cross-polytope LSH, which is used in `FALCONN` [44], was originally proposed as a fast heuristic method with no proven asymptotic guarantees [46]; only later was it discovered that this method is theoretically superior to other methods as well [3, 32], and can be made even more practical. Recently also for hypercube LSH [46] improved theoretical guarantees were obtained [35], and the heuristic LSH forest approach [10] was also "made theoretical" with provable performance guarantees in high dimensions [7]. The goals of a theoretical analysis of graph-based methods are therefore twofold: to understand their asymptotic performance better, and to find ways to further improve these methods in practice.

1.1 Contributions

We take a first step towards a better theoretical understanding of graph-based approaches for the (approximate) nearest neighbor problem, by rigorously analyzing the asymptotic performance of the basic greedy nearest neighbor graph approach. We show that for random data sets on the Euclidean unit sphere and for arbitrary approximation factors $c > 1$, this method provably achieves asymptotic query exponents $\rho < 1$. We further show how to obtain efficient time–space trade-offs, by making the related near neighbor graph either more connected (more space, better query time) or less connected (less space, worse query time).

Sparse data sets. In the case the data set on the unit sphere[1] has size $n = 2^{o(d)}$, and an unusually near neighbor lies at distance $r = \sqrt{2}/c$, the trade-offs we obtain for finding such a close vector to a random query vector are governed by the following relation.

▶ **Theorem 1** (Time–space trade-offs for sparse data). *For $c > 1$ and $\rho_q, \rho_s \geq 0$ satisfying*

$$(2c^2 - 1)\rho_q + 2c^2(c^2 - 1)\sqrt{\rho_s(1 - \rho_s)} \geq c^4, \tag{2}$$

there exists a graph-based (c, r)-ANN data structure for data sets of size $n = 2^{o(d)}$ on the sphere using space $n^{1+\rho_s+o(1)}$ and query time $n^{\rho_q+o(1)}$.

Minimizing the query complexity in this asymptotic analysis corresponds to setting $\rho_s = \frac{1}{2}$, in which case $\rho_q = c^2/(2c^2 - 1)$. Note that, unlike in various hash-based constructions, there is a natural limit to the maximum space complexity of graph-based approaches: we cannot store more than n neighbors per vertex. Our analysis however suggests that when doing a greedy search in the graph, storing more than $\sqrt{n}$ neighbors per vertex does not further improve the query complexity, compared to starting over at a random new vertex.

To compare the above trade-offs with hash-based results, recall that in [5] the authors obtained hash-based time–space trade-offs defined by the following inequality:

$$c^2\sqrt{\rho_q} + (c^2 - 1)\sqrt{\rho_s} \geq \sqrt{2c^2 - 1}. \tag{3}$$

Although in most cases the optimal[2] hash-based trade-offs from (3) are strictly superior to the graph-based trade-offs from Theorem (1), we remark that in the regime of small approximation factors $c \approx 1$ and near-linear space $\rho_s \approx 0$, i.e. when there is no unusually near neighbor and we wish to use only slightly more space than is required for storing the input list, both inequalities above translate to:

$$\rho_q = 1 - 4(c - 1)\sqrt{\rho_s} \cdot (1 + o(1)). \tag{4}$$

Here $o(1)$ vanishes as $\rho_s \to 0$ and $c \to 1$. So for truly random instances without any planted, unusually near neighbors, and when using a limited amount of memory, graph-based methods are asymptotically equally powerful for finding (approximate) nearest neighbors as optimal hash-based methods. In other words, in this regime graph-based approaches will remain competitive with hash-based methods even when the data sets become very large, and the dimensionality further increases.

---

[1] Note that the near neighbor problem on the Euclidean unit sphere is of particular interest due to the reduction from the near neighbor problem in all of $\mathbb{R}^d$ (under the ℓ_2-norm) to solving the spherical case [6]. Furthermore, using standard reductions the results for the ℓ_2-norm translate to results for $\mathbb{R}^d$ with the ℓ_1-norm as well.

[2] Within a certain probing model, the hash-based trade-offs from (3) were proven to be optimal in [5].

On the negative side, our analysis suggests that when there is an unusually near (planted) neighbor to the query, or when we are able to use much more space than the amount required to store the data set, the best known hash-based approaches are superior to the basic graph-based method analyzed in this paper. This could be caused either by our analysis not being tight, the considered algorithm not being as optimized, or due to graph-based approaches simply not being able to profit as much from such unusual circumstances. Note again that hash-based approaches are able to effectively use very large amounts of space, with a large number of fine-grained hash tables/partitions, whereas graph-based methods seem limited by using at most n^2 space. We therefore conjecture that the worse asymptotic performance for "unusual" problem instances is inherent to graph-based methods.

Dense data sets. For settings where the data set consists of $n = 2^{\Theta(d)}$ uniformly random points from the unit sphere, the asymptotic performance of near neighbor searching depends not only on the distance to the (planted) nearest neighbor, but also on the density of the data set, i.e. the relation between d and n. Motivated by data sets appearing in practice, we concretely analyze the case $(\log n)/d \approx 1/5$ and show that the resulting trade-offs for the exact nearest neighbor problem without planted neighbors are significantly better than hyperplane LSH [18]; comparable to or better than cross-polytope LSH [3, 44] and spherical cap LSH [4]; but slightly worse than spherical LSF [11, 5]. The superior asymptotic performance compared to `FALCONN` for this setting suggests that graph-based approaches will remain competitive with the most practical hash-based approaches, even in very high dimensions.

Future work. Various open problems remain to obtain a better theoretical (and practical) understanding of different near neighbor techniques, in particular related to graph-based approaches. We state some remaining open problems below:

- Although our analysis for "small steps" (see the full version) is tight, we assumed that afterwards we either immediately find the planted nearest neighbor as one of the neighbors in the graph, or we fail and start over from a new random node. This seems rather pessimistic, and perhaps the complexities can be further improved with a tighter analysis, without changing the underlying algorithm or graph construction.
- Other heuristic graph-based approaches appear to perform even better in practice, and an open problem is to rigorously analyze their asymptotic behavior as well.
- A practical drawback of using nearest neighbor graphs is that updating the data structure (in particular: inserting new data points) can be rather costly, especially in comparison with hash-based constructions. To make the data structure both efficient and dynamic, one would ideally obtain better bounds on the insertion complexity as well.
- In hash-based literature, the case of sparse data sets has arguably almost been "solved" with upper bounds matching lower bounds (within a certain model). Is it possible to find similar lower bounds for graph-based near neighbor searching?
- Hash-based and graph-based approaches could be considered complementary solutions to the same problem, as worst-case problem instances for one approach are best-case instances for the other. Can both techniques be combined to obtain even better results?

Outline. The remainder of this paper is organized as follows. In Section 2, we introduce preliminary results and notation. Section 3 describes problem instances considered in this paper. Section 4 analyzes the complexities of finding near neighbors with a greedy graph search, and Sections 5 and 6 consider asymptotics for sparse and dense data sets, respectively.

2 Preliminaries

2.1 Notation

Let $\|\cdot\|$ denote the Euclidean norm, and let $\langle\cdot,\cdot\rangle$ denote the standard dot product. We write $\mathcal{S}^{d-1} = \{\boldsymbol{u} \in \mathbb{R}^d : \|\boldsymbol{u}\| = 1\}$ for the Euclidean unit sphere in $\mathbb{R}^d$. We write $X \sim \chi(\mathcal{X})$ to denote that the random variable X is sampled from the probability distribution χ over the set $\mathcal{X}$, and we write $\mathcal{U}(\mathcal{X})$ for the uniform distribution over $\mathcal{X}$. Given a vector $\boldsymbol{x}$ (written in boldface), we further write $x_i = \langle \boldsymbol{x}, \boldsymbol{e}_i\rangle$ for the ith coordinate of $\boldsymbol{x}$.

We denote directed graphs by $G = (V, A)$ where V denotes the set of vertices, and $A \subseteq V \times V$ denotes the set of directed arcs. A directed graph is called symmetric if $(v_1, v_2) \in A$ iff $(v_2, v_1) \in A$. Symmetric directed graphs can also be viewed as undirected graphs $G = (V, E)$ where edges are unordered subsets of V of size 2.

2.2 Geometry on the sphere

Let $\mathcal{C}_{\boldsymbol{x},\alpha} = \{\boldsymbol{u} \in \mathcal{S}^{d-1} : \langle \boldsymbol{u}, \boldsymbol{x}\rangle \geq \alpha\}$ denote the spherical cap centered at $\boldsymbol{x} \in \mathcal{S}^{d-1}$ of "height" $\alpha \in (0, 1)$, and let $C(\alpha)$ denote its volume relative to the entire unit sphere. Let $\mathcal{W}_{\boldsymbol{x},\alpha,\boldsymbol{y},\beta} = \mathcal{C}_{\boldsymbol{x},\alpha} \cap \mathcal{C}_{\boldsymbol{y},\beta}$ with $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{S}^{d-1}$ and $\alpha, \beta \in (0, 1)$ denote the intersection of two spherical caps, and let its volume relative to the volume of the unit sphere be denoted $W(\alpha, \beta, \gamma)$, where $\gamma = \langle \boldsymbol{x}, \boldsymbol{y}\rangle$ is the cosine of the angle between $\boldsymbol{x}$ and $\boldsymbol{y}$. We will also call the latter objects *wedges*. The volumes of these objects correspond to probabilities on the sphere as follows:

$$C(\alpha) = \mathbb{P}_{\boldsymbol{X}\sim\mathcal{U}(\mathcal{S}^{d-1})}(X_1 > \alpha), \tag{5}$$

$$W(\alpha, \beta, \gamma) = \mathbb{P}_{\boldsymbol{X}\sim\mathcal{U}(\mathcal{S}^{d-1})}(X_1 > \alpha, X_1\gamma + X_2\sqrt{1-\gamma^2} > \beta). \tag{6}$$

The volumes of spherical caps and wedges can be estimated as follows (see e.g. [11, 5]).

▶ **Lemma 2** (Volume of a spherical cap). *Let $\alpha \in (0, 1)$. Then:*

$$C(\alpha) = d^{\Theta(1)} \cdot (1-\alpha^2)^{d/2}. \tag{7}$$

▶ **Lemma 3** (Volume of a wedge). *Let $\alpha, \beta, \gamma \in (0, 1)$. Then:*

$$W(\alpha, \beta, \gamma) = d^{\Theta(1)} \cdot \begin{cases} \left(\dfrac{1-\alpha^2-\beta^2-\gamma^2+2\alpha\beta\gamma}{1-\gamma^2}\right)^{d/2} & \text{if } 0 < \gamma \leq \min\{\frac{\alpha}{\beta}, \frac{\beta}{\alpha}\}; \\ (1-\alpha^2)^{d/2} & \text{if } \frac{\beta}{\alpha} \leq \gamma < 1; \\ (1-\beta^2)^{d/2} & \text{if } \frac{\alpha}{\beta} \leq \gamma < 1. \end{cases} \tag{8}$$

For the wedge, the volume can alternatively be described (up to polynomial factors in d) as the volume of a spherical cap with height $\delta = \sqrt{\frac{\alpha^2+\beta^2-2\alpha\beta\gamma}{1-\gamma^2}}$. In Figure 1, this δ corresponds to the smallest distance from points in $\mathcal{W}_{\boldsymbol{x},\alpha,\boldsymbol{y},\beta}$ to the origin, i.e. the distance from the intersection of the blue and red lines in this projection of the sphere to the origin. In case $\gamma \geq \frac{\alpha}{\beta}$ with $\alpha \leq \beta$, this distance from the origin becomes $\delta = \beta$ and therefore $W(\alpha, \beta, \gamma) = d^{\Theta(1)} \cdot C(\beta)$. In that case, one essentially has $\mathcal{C}_{\boldsymbol{x},\alpha} \cap \mathcal{C}_{\boldsymbol{y},\beta} \approx \mathcal{C}_{\boldsymbol{y},\beta}$.

We will be using various properties of these wedges, and we state some of them below.

▶ **Lemma 4** (Wedge properties). *For $\alpha, \beta, \gamma \in (0, 1)$ with $\gamma < \min\{\frac{\alpha}{\beta}, \frac{\beta}{\alpha}\}$:*

1. *For d sufficiently large, $W(\alpha, \beta, \gamma)$ is decreasing with α, β and increasing with γ;*
2. *For d sufficiently large, $W(\alpha, \beta, \beta)$ is decreasing with β.*

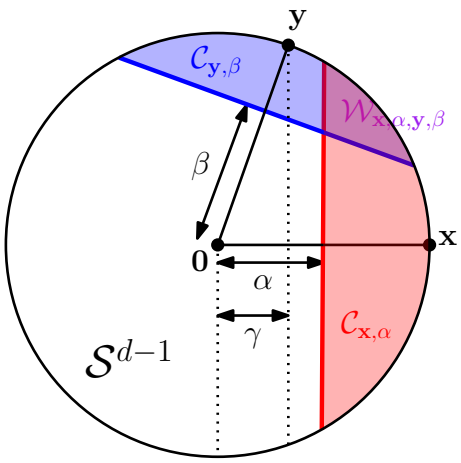


Figure 1 Geometry on the sphere. The relative volume of the wedge, $\mu(\mathcal{W}_{\boldsymbol{x},\alpha,\boldsymbol{y},\beta})/\mu(\mathcal{S}^{d-1})$, is denoted $W(\alpha,\beta,\gamma)$, with γ (as in the sketch) denoting the cosine of the angle between $\boldsymbol{x}$ and $\boldsymbol{y}$.

Proof. For 1. the result follows by taking derivatives w.r.t. α,β,γ of the "asymptotic part" of W (i.e. ignoring the $d^{\Theta(1)}$ term), and observing that these derivatives are always negative, negative, and positive respectively. For 2. we take the derivative w.r.t. β of $W(\alpha,\beta,\beta)$ and observe that this derivative is always negative. ◀

The following lemma further describes that if we add a small amount of "slack" to one of the variables, then the volume of the resulting wedge will still be very similar to the volume of the wedge with the original parameters – if the slack is small, then we only lose at most a polynomial factor in the volume.

▶ **Lemma 5** (Polynomial slack). *For fixed $\alpha,\beta,\gamma \in (0,1)$ with $\gamma < \min\{\frac{\alpha}{\beta},\frac{\beta}{\alpha}\}$ we have*

$$W(\alpha,\beta\pm\tfrac{1}{d},\gamma) = d^{\mp\Theta(1)}\cdot W(\alpha,\beta,\gamma), \qquad \text{and } W(\alpha,\beta,\gamma\pm\tfrac{1}{d}) = d^{\pm\Theta(1)}W(\alpha,\beta,\gamma). \quad (9)$$

Proof. This follows from writing out the left hand sides, pulling out the factors $W(\alpha,\beta,\gamma)$, and noting that the remaining factor $(1\pm\varepsilon_d)^d$ for some expression ε_d is at most polynomial in d, due to the conditions on γ and α,β,γ being constant in d. ◀

3 Random instances

For the graph-based approach in this paper, where points are connected to their nearest neighbor and we perform a walk on this graph starting from a random node, it is impossible to solve worst-case instances of (approximate) nearest neighbor searching.

▶ **Proposition 6** (Worst-case data sets). *For near neighbor graph approaches, where (i) each vertex is only connected to a number of its nearest neighbors, and (ii) queries are answered by performing a greedy search on this graph to obtain better estimates, it is impossible to solve worst-case (approximate) near neighbor instances in sub-linear time.*

Proof. As a potential worst-case problem instance to such a strategy, consider a query $\boldsymbol{q} \approx \boldsymbol{e}_1$, a planted nearest neighbor $\boldsymbol{p}^* \approx \boldsymbol{e}_1$ close to the query, and let all other points $\boldsymbol{p} \in \mathcal{D}\setminus\{\boldsymbol{p}^*\}$ satisfy $\boldsymbol{p} \approx -\boldsymbol{e}_1$. Then the preprocessed near neighbor graph will consist of a large connected component containing the points $\mathcal{D}\setminus\{\boldsymbol{p}^*\}$, and an isolated vertex $\boldsymbol{p}^*$, which may have outgoing edges, but has no mutual friends. With probability $1-1/n$, starting at a

random vertex and performing a walk on this (directed) graph will therefore not yield the true nearest neighbor $\boldsymbol{p}^*$, while all other vertices are only approximate solutions with very high approximation factors; by essentially setting $\boldsymbol{p}^* = \boldsymbol{q}$, we can guarantee that even no reasonable approximate solution will be found. ◀

Although it may be possible to tweak the algorithm and/or the underlying graph so that even such worst-case instances can still be solved efficiently, we will therefore focus on average-case, random instances defined below.

Throughout, we will assume that points $\boldsymbol{p} \in \mathcal{D}$ are independently and uniformly distributed on the unit sphere, except for (potentially) a planted nearest neighbor $\boldsymbol{p}^* \in \mathcal{D}$ which lies very close to the query vector $\boldsymbol{q}$. Alternatively, this can be interpreted as taking a uniformly random $\boldsymbol{p}^* \sim \mathcal{U}(\mathcal{D})$, and choosing the query vector as $\boldsymbol{q} = \boldsymbol{p}^* + \boldsymbol{e}$ for a short error vector $\boldsymbol{e}$. Given such a problem instance, we wish to recover $\boldsymbol{p}^*$ with non-negligible probability.

Note that the near neighbor problem on the Euclidean unit sphere, as considered here, is of special interest as such a solution allows one to the Euclidean near neighbor problem in all of $\mathbb{R}^d$ using techniques of Andoni–Razenshteyn [6]. Furthermore, it is well-known that using standard reductions, the results for the ℓ_2-norm translate to results for $\mathbb{R}^d$ with the ℓ_1-norm as well. Efficient algorithms for the unit sphere therefore translate to solutions for many other problems as well.

For $\boldsymbol{q} \sim \mathcal{U}(\mathcal{D})$, and data sets following a uniformly random distribution, with overwhelming probability the (non-planted) second nearest neighbor $\boldsymbol{p} \in \mathcal{D} \setminus \{\boldsymbol{p}^*\}$ to $\boldsymbol{q}$ has inner product $\langle \boldsymbol{p}, \boldsymbol{q} \rangle = \mu(1 + o(1))$ and lies at distance $\|\boldsymbol{q} - \boldsymbol{p}\| = \sqrt{2(1-\mu)}(1 + o(1))$ from $\boldsymbol{q}$, with μ satisfying:

$$\mu = \sqrt{1 - n^{-2/d}}. \qquad \text{(Alternatively: } n \approx 1/C(\mu).\text{)} \tag{10}$$

The natural scenario to consider for random ANN instances is then to let $c \cdot r = \mu$, so that the single planted nearest neighbor lies at distance $r = \mu/c$ from the target, and so that this planted near neighbor lies a factor c closer to $\boldsymbol{q}$ than all other points in the data set. These problem instances were also studied in for example [3, 5].

Sparse data sets. We will refer to data sets of size $n = 2^{o(d)}$ (in other words: $d = \omega(\log n)$) as *sparse* data sets. For these instances, the expected (non-planted) nearest neighbor distance is $\mu = \sqrt{2} + o(1)$, and the planted nearest neighbor which we wish to recover therefore lies at distance $r = \sqrt{2}/c\,(1 + o(1))$ from the target. Note that the case $n = 2^{o(d/\log d)}$ can always be reduced to $n = 2^{\Theta(d/\log d)}$ through a random projection onto a lower-dimensional space, approximately maintaining all pairwise distances between points in the data set [31].

Dense data sets. We will refer to data sets of size $n = 2^{\Theta(d)}$ ($d = \Theta(\log n)$) as *dense* data sets. In this case, with overwhelming probability the nearest neighbor to a randomly chosen query point $\boldsymbol{q}$ on the sphere will lie at distance $c \cdot r = \mu < \sqrt{2}$ from $\boldsymbol{q}$, and for (c, r)-ANN the planted nearest neighbor would therefore lie at distance $r = \mu/c$. For the limiting case of $c \to 1$, we wish to recover a vector at distance μ. Note that the "curse of dimensionality" [30] does not necessarily apply to average-case dense data sets – finding exact nearest neighbors for random dense data sets can often be done in sub-linear time [34, 11].

4 Graph-based near neighbor searching

4.1 Algorithm description

The nearest neighbor graph and corresponding near neighbor search algorithm we will analyze are very similar to a greedy search in the k-nearest neighbor graph, i.e. very similar to the approach of `KGraph`. Recall that the directed k-nearest neighbor graph $G = (V, A)$ has vertex set $V = \mathcal{D}$, and an arc runs from $\boldsymbol{p}$ to $\boldsymbol{p}'$ iff $\boldsymbol{p}'$ belongs to the k closest points to $\boldsymbol{p}$ in $\mathcal{D}$. Given a query $\boldsymbol{q}$, searching for a nearest vector in this graph is commonly done as outlined in Algorithm 1, with $\varepsilon = 0$, and with $\mathcal{B}(\boldsymbol{p})$ denoting the set of neighbors to $\boldsymbol{p}$ in this graph. Note that for large $k = n^{\Theta(1)}$, this graph is almost symmetric. The condition of belonging to the k nearest neighbors is however somewhat impractical to work with analytically, and so we will use the following slightly different graph (and search algorithm) instead.

▶ **Definition 7** (The α-near neighbor graph)**.** Let $\alpha \in (0, 1)$, and let $\mathcal{D} \subset \mathcal{S}^{d-1}$. We define the α-near neighbor graph as the undirected graph $G = (V, E)$ with vertex set $V = \mathcal{D}$, and with an edge between $\boldsymbol{p}, \boldsymbol{p}' \in E$ if and only if $\langle \boldsymbol{p}, \boldsymbol{p}' \rangle \geq \alpha$.

The parameter α roughly corresponds to (a function of) k: large α correspond to small k, and small α to large k. Asymptotically, the approximate relation between k and α can be stated through the following simple relation $k \approx n \cdot C(\alpha)$. The main difference is that rather than fixing k and varying the required distance between points for an edge, we fix a bound on the distance between two connected points in the graph, and therefore we will have slight variations in the number of neighbors k from vertex to vertex. Notice the similarity with spherical cap LSH [4] and in particular spherical LSF [11, 5]: we essentially use n random spherical filters centered around our data points, and we add vectors to filter buckets the same way as in spherical LSF: if the inner product with the filter vector $\boldsymbol{p}$ is sufficiently large, we add the point to this bucket $\mathcal{B}(\boldsymbol{p})$. However, in the spirit of graph-based approaches we search for a path on the nearest neighbor graph that ends at the nearest neighbor as in Algorithm 1, rather than checking a number of filters close to the query point to see if the nearest neighbor is contained in any of those.

To process a query $\boldsymbol{q}$, we first sample a uniformly random point $\boldsymbol{p} \sim \mathcal{U}(\mathcal{D})$. Then we go through its neighbors $\mathcal{B}(\boldsymbol{p})$ to see if any of these vectors $\boldsymbol{p}' \in \mathcal{B}(\boldsymbol{p})$ are closer to $\boldsymbol{q}$ than $\boldsymbol{p}$. If so, we use this as our new $\boldsymbol{p} \leftarrow \boldsymbol{p}'$, and we again see if any of its neighbors are closer to $\boldsymbol{q}$. We repeat this procedure until no more vectors in $\mathcal{B}(\boldsymbol{p})$ are closer to $\boldsymbol{q}$ than $\boldsymbol{p}$ itself, in which case $\boldsymbol{p}$ is our estimate for the real nearest neighbor $\boldsymbol{p}^*$ to $\boldsymbol{q}$. This may ($\boldsymbol{p} = \boldsymbol{p}^*$) or may not ($\boldsymbol{p} \neq \boldsymbol{p}^*$) actually be the true nearest neighbor to $\boldsymbol{q}$, and to obtain a higher success rate we repeat the process several times, each time starting from a random point $\boldsymbol{p} \sim \mathcal{U}(\mathcal{D})$.

While the focus is on minimizing the query time, Algorithms 2–3 also demonstrate how to do updates to this data structure, when vectors are inserted in $\mathcal{D}$ or removed from $\mathcal{D}$. These parts of the algorithm may well be improved upon, and an important open question is to make updates (in particular insertions) as efficient as partition-based methods, such as in [5].

4.2 High-level proof description

To obtain asymptotic complexities for this approach for the (approximate) nearest neighbor problem, the following statements (for some value γ_{max}) are proven in the full version:

- **Small steps:** If $\boldsymbol{p} \in \mathcal{D}$ satisfies $\langle \boldsymbol{p}, \boldsymbol{q} \rangle \ll \gamma_{max}$, then with non-negligible probability $d^{-\Theta(1)}$ we will find a slightly nearer neighbor $\boldsymbol{p}' \in \mathcal{B}(\boldsymbol{p})$ to $\boldsymbol{q}$.
- **Giant leap:** If $\boldsymbol{p} \in \mathcal{D}$ satisfies $\langle \boldsymbol{p}, \boldsymbol{q} \rangle \approx \gamma_{max}$, the probability of immediately finding the exact nearest neighbor $\boldsymbol{p}^*$ in the bucket $\mathcal{B}(\boldsymbol{p})$ is larger than some given bound.

Algorithm 1 α-NN graph: QUERY($\boldsymbol{q}$).

```
p ~ U(D)        // random starting point
while ⟨p, q⟩ < γ* do        // γ* = ⟨p*, q⟩
    progress ← false
    for each p' ∈ B(p) do
        if ⟨p', q⟩ ≥ ⟨p, q⟩ + 1/d then
            p ← p'        // nearer neighbor
            progress ← true
    if not progress then
        p ~ U(D)        // start over
return p
```

Algorithm 2 α-NN graph: INSERT($\boldsymbol{p}$).

```
B(p) ← ∅        // new bucket
for each p' ∈ D do
    if ⟨p, p'⟩ ≥ α then
        B(p) ← B(p) ∪ {p'}
        B(p') ← B(p') ∪ {p}
```

Algorithm 3 α-NN graph: DELETE($\boldsymbol{p}$).

```
for each p' ∈ B(p) do
    B(p') ← B(p') \ {p}
B(p) ← ∅        // delete bucket
```

Figure 2 Algorithms for querying the α-near neighbor graph with a query point $\boldsymbol{q}$, and for inserting/deleting points from this data structure. Here γ^* is the (expected) inner product between $\boldsymbol{q}$ and its true nearest neighbor. Initializing the data structure is done by choosing $\alpha \in (0, 1)$ and for instance calling INSERT($\boldsymbol{p}$) for all $\boldsymbol{p} \in \mathcal{D}$ to construct the graph adjacency buckets $\mathcal{B}(\boldsymbol{p})$.

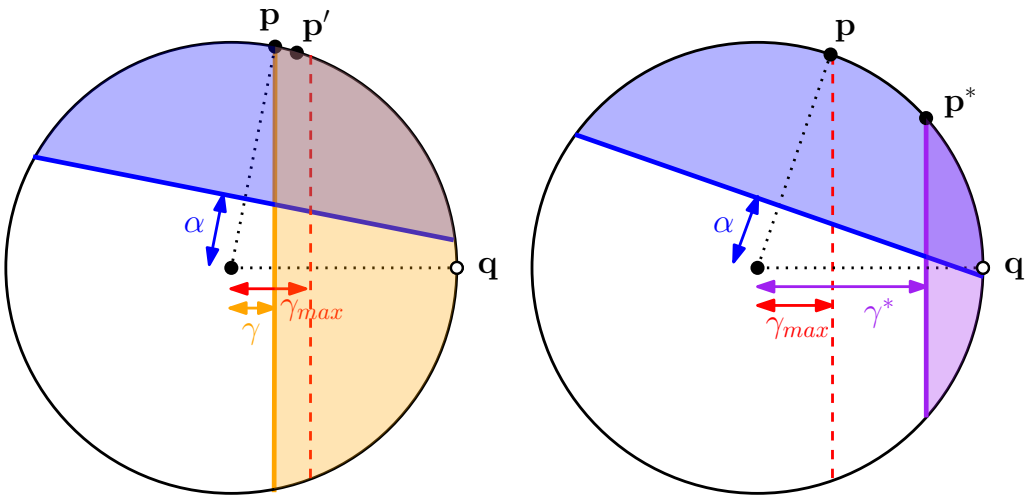


Figure 3 A sketch of the analyses for small steps (left) and the giant leap (right). The point $\boldsymbol{p} \in \mathcal{D}$ denotes the current near neighbor estimate for the query $\boldsymbol{q}$, and we want to make progress either by finding a *slightly nearer* neighbor $\boldsymbol{p}' \in \mathcal{B}(\boldsymbol{p})$ when $\boldsymbol{p}$ is *far away* from $\boldsymbol{q}$ (left), or finding the *exact* nearest neighbor $\boldsymbol{p}^* \in \mathcal{B}(\boldsymbol{p})$ to the query $\boldsymbol{q}$ when $\boldsymbol{p}$ is already quite close to $\boldsymbol{q}$ (right). The threshold separating these two cases is γ_{max}.

- **Small steps (left).** If the current near neighbor $\boldsymbol{p}$ has inner product $\langle \boldsymbol{p}, \boldsymbol{q} \rangle = \gamma \ll \gamma_{max}$ with $\boldsymbol{q}$, then we expect that several points $\boldsymbol{p}' \in \mathcal{D} \setminus \{\boldsymbol{p}^*\}$ still exist which lie (slightly) closer to $\boldsymbol{q}$ than $\boldsymbol{p}$, and we hope at least one of them is an α-near neighbor to $\boldsymbol{p}$ as well. Since a nearer neighbor in $\mathcal{B}(\boldsymbol{p})$ to $\boldsymbol{q}$ by definition lies in $\mathcal{W}_{\boldsymbol{p},\alpha,\boldsymbol{q},\langle \boldsymbol{p},\boldsymbol{q}\rangle}$ (the intersection of the solid spherical caps), and the data set is assumed to be uniformly random on the sphere, the probability of finding at least one such nearer neighbor is proportional to $n \cdot W(\alpha, \gamma, \gamma)$.
- **Giant leap (right).** Once we find a near neighbor $\boldsymbol{p}$ to $\boldsymbol{q}$ with inner product $\langle \boldsymbol{p}, \boldsymbol{q} \rangle \approx \gamma_{max}$, we would like to show that in the next step, we will find $\boldsymbol{p}^* \in \mathcal{B}(\boldsymbol{p})$ with a certain (small) probability. Assuming $\boldsymbol{p}^*$ is uniformly distributed on $\mathcal{C}_{\boldsymbol{q},\gamma^*}$, this corresponds to the probability that $\boldsymbol{p}^* \in \mathcal{W}_{\boldsymbol{p},\gamma_{max},\boldsymbol{q},\gamma^*}$ (the intersection of the solid spherical caps), conditioned on the event $\boldsymbol{p}^* \in \mathcal{C}_{\boldsymbol{q},\gamma^*}$ (the rightmost spherical cap). This probability is therefore proportional to $W(\alpha, \gamma^*, \gamma_{max})/C(\gamma^*)$.

- **Randomization:** Since "giant leaps" may often fail, we argue that starting over at random nodes a sufficiently large number of times leads to a constant success probability.
- **Encountered vertices:** We prove that the number of vertices in each bucket, and on each walk through the graph, can be bounded by small multiples of their expected values.
- **Query complexity:** Using these results, we derive bounds on the time complexity for answering queries, with and without starting over at random nodes in the graph.
- **Space complexity:** Similarly, since the number of edges in the graph can be bounded appropriately, we obtain tight bounds on the required space complexity.
- **Update complexities:** Finally, we analyze what are the (naive) costs for updating the data structure (inserting/deleting points).

Together, these results lead to the following main result, stating exactly what are the costs for finding near neighbors. Unless stated otherwise, the "time" complexity corresponds to the *query* time complexity.

▶ **Theorem 8** (Near neighbor costs). *Using the α-near neighbor graph with $\alpha \in (0,1)$ and $nC(\alpha) \gg 1$, and using Algorithms 1–3, we can solve the planted near neighbor problem with the following costs, with $\gamma_{max} = (1+o(1))\sqrt{\frac{\mu^2-\alpha^2}{1-2\alpha+\mu^2}}$:*

$$\texttt{Time} = d^{O(1)} nC(\alpha)C(\gamma^*)/W(\alpha,\gamma^*,\gamma_{max}), \tag{11}$$

$$\texttt{Space} = O(n^2 C(\alpha)\log n), \tag{12}$$

$$\texttt{Insert} = O(dn), \tag{13}$$

$$\texttt{Delete} = O(nC(\alpha)\log(nC(\alpha))). \tag{14}$$

5 Asymptotics for sparse data sets

Due to space limitations, the derivation of the asymptotics on the various costs of graph-based near neighbor searching for large d can be found in the full version only. Its derivation mainly consists of carefully writing out the costs stated in the previous theorem, and doing series expansions for $\mu = \sqrt{1-n^{-2/d}} = o(1)$.

▶ **Theorem 9** (Complexities for sparse data). *Let $n = 1/C(\mu) = 2^{o(d)}$ and let $\alpha = \kappa \cdot \mu$. Let $\gamma^* = 1 - 1/c^2$ denote the inner product between the query and the (planted) nearest neighbor. Using the α-NNG with $\alpha = \kappa\cdot\mu$ with $\kappa \in (\sqrt{(\gamma^*)^2/(1+(\gamma^*)^2)}, 1)$, with high probability we can solve the sparse near neighbor problem in several iterations with the following complexities:*

$$\texttt{Time} = n^{\left\lceil \frac{1-2\gamma^*\sqrt{\kappa^2(1-\kappa^2)}}{1-(\gamma^*)^2} + o(1) \right\rceil}, \tag{15}$$

$$\texttt{Space} = n^{2-\kappa^2+o(1)}, \tag{16}$$

$$\texttt{Insert} = n^{1+o(1)}, \tag{17}$$

$$\texttt{Delete} = n^{1-\kappa^2+o(1)}. \tag{18}$$

Denoting the query time complexity by $\texttt{Time} = n^{\rho_q+o(1)}$ and the space complexity by $\texttt{Space} = n^{1+\rho_s+o(1)}$, the trade-off between these costs can be expressed using the following inequality:

$$(2c^2-1)\rho_q + 2c^2(c^2-1)\sqrt{\rho_s(1-\rho_s)} \geq c^4. \tag{19}$$

As described in the introduction, for $c \approx 1$ and $\rho_s \to 0$, the above condition on ρ_q scales as $\rho_q \geq 1 + 4(c-1)^2 + O((c-1)^4)$, which is equivalent to the scaling near $c = 1$ for the optimal

partition-based near neighbor method of [5]. For larger c and when using more space, this trade-off is not better than the best hash-based methods – by substituting $\sqrt{\rho_s(1-\rho_s)} \leq \frac{1}{2}$, we obtain the necessary (but not sufficient) condition $\rho_q \geq c^2/(2c^2-1)$, which shows that the query complexity never reduces beyond $\sqrt{n}$, even for large c. This inability to "profit" from large approximation factors may be inherent to graph-based approaches, or at least to the greedy graph-based approach considered in this paper.

For the "balanced" trade-off of $\rho_q = \rho_s = \rho$, the condition on the exponents translates to $\rho \geq c^4/(2c^4 - 2c^2 + 1)$. For small $c \approx 1$, this leads to the asymptotic scaling $\rho = 1 - 4(c-1)^2 + O((c-1)^3)$, which is slightly worse than the optimal hash-based trade-offs of [4, 5]: $\rho = 1/(2c^2-1) = 1 - 4(c-1) + 14(c-1)^2 + O((c-1)^3)$. Also note again the asymptotics of $\rho \to \frac{1}{2}$ for large c for graph-based methods,

6 Asymptotics for dense data sets

For data sets of size $n = 2^{\Theta(d)}$, the asymptotics from the previous section do not apply; these assumed that $\mu = \sqrt{1-n^{-2/d}} = o(1)$. In some applications the relation $d = \Theta(\log n)$ is more accurate, and arguably even if e.g. $d = \Theta(\log n \log\log n)$ grows faster than $\log n$, asymptotics for the sparse regime may be rather optimistic; $\log\log n$ terms are then considered "large", even though for realistic parameters $\log\log n$ may just as well be considered constant.

In the full version, we performed a case study for an application where $\mu = \sqrt{1-n^{-2/d}} = \frac{1}{2}$, so that $n \approx 2^{d/5}$. This example is motivated by algorithms for solving hard lattice problems and cryptanalyzing lattice-based cryptosystems, which have previously been improved using other (hash-based) near neighbor methods [33, 12, 11]. Comparable results may hold for other applications as well; the GloVe data set [41] contains $n = 1.2M \approx 2^{20}$ vectors in $d = 100$ dimensions, which corresponds to $n \approx 2^{d/5}$, while the SIFT features data set [1] has $n = 1M$ vectors in $d = 128$ dimensions, corresponding to $n \approx 2^{0.35d}$. The conclusions regarding the comparison of graph-based near neighbor searching techniques with hash-based methods may therefore apply to such data sets as well.

References

1 Laurent Amsaleg and Hervé Jégou. Datasets for approximate nearest neighbor search, 2010. URL: http://corpus-texmex.irisa.fr/.

2 Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *FOCS*, pages 459–468, 2006. doi:10.1109/FOCS.2006.49.

3 Alexandr Andoni, Piotr Indyk, Thijs Laarhoven, Ilya Razenshteyn, and Ludwig Schmidt. Practical and optimal LSH for angular distance. In *NIPS*, pages 1225–1233, 2015. URL: https://papers.nips.cc/paper/5893-practical-and-optimal-lsh-for-angular-distance.

4 Alexandr Andoni, Piotr Indyk, Huy Lê Nguyên, and Ilya Razenshteyn. Beyond locality-sensitive hashing. In *SODA*, pages 1018–1028, 2014. doi:10.1137/1.9781611973402.76.

5 Alexandr Andoni, Thijs Laarhoven, Ilya Razenshteyn, and Erik Waingarten. Optimal hashing-based time-space trade-offs for approximate near neighbors. In *SODA*, pages 47–66, 2017. doi:10.1137/1.9781611974782.4.

6 Alexandr Andoni and Ilya Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *STOC*, pages 793–801, 2015. doi:10.1145/2746539.2746553.

7 Alexandr Andoni, Ilya Razenshteyn, and Negev Shekel Nosatzki. LSH forest: Practical algorithms made theoretical. In *SODA*, pages 67–78, 2017. URL: `https://dl.acm.org/citation.cfm?id=3039691`.

8 Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. In *SODA*, pages 573–582, 1994. URL: `http://dl.acm.org/citation.cfm?id=314464.314652`.

9 Martin Aumueller, Erik Bernhardsson, and Alexander Faithfull. ANN benchmarks, 2017. URL: `http://sss.projects.itu.dk/ann-benchmarks/`.

10 Mayank Bawa, Tyson Condie, and Prasanna Ganesan. LSH forest: self-tuning indexes for similarity search. In *WWW*, pages 651–660, 2005.

11 Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In *SODA*, pages 10–24, 2016. `doi:10.1137/1.9781611974331.ch2`.

12 Anja Becker and Thijs Laarhoven. Efficient (ideal) lattice sieving using cross-polytope LSH. In *AFRICACRYPT*, pages 3–23, 2016. `doi:10.1007/978-3-319-31517-1_1`.

13 Erik Bernhardsson. ANN benchmarks, 2016. URL: `https://github.com/erikbern/ann-benchmarks`.

14 Erik Bernhardsson. ANNOY, 2017. URL: `https://github.com/spotify/annoy`.

15 Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, 2006.

16 Leonid Boytsov and Bilegsaikhan Naidan. NMSLib, 2017. URL: `https://github.com/searchivarius/nmslib`.

17 M.R. Brito, E.L. Chavez, A.J. Quiroz, and J.E. Yukich. Connectivity of the mutual k-nearest-neighbor graph in clustering and outlier detection. *Statistics & Probability Letters*, 35(1):33–42, 1997.

18 Moses S. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, pages 380–388, 2002. `doi:10.1145/509907.509965`.

19 Jie Chen, Haw-ren Fang, and Yousef Saad. Fast approximate kNN graph construction for high dimensional data via recursive Lanczos bisection. *Journal of Machine Learning Research*, 10(Sep):1989–2012, 2009. URL: `https://dl.acm.org/citation.cfm?id=1755852`.

20 Tobias Christiani. A framework for similarity search with space-time tradeoffs using locality-sensitive filtering. In *SODA*, pages 31–46, 2017. `doi:10.1137/1.9781611974782.3`.

21 Kenneth L. Clarkson. Nearest neighbor queries in metric spaces. In *STOC*, pages 609–617, 1997. `doi:10.1145/258533.258655`.

22 Michael Connor and Piyush Kumar. Fast construction of k-nearest neighbor graphs for point clouds. *IEEE Transactions on Visualization and Computer Graphics*, 16(4):599–608, 2010. `doi:10.1109/TVCG.2010.9`.

23 Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *SOCG*, pages 253–262, 2004. `doi:10.1145/997817.997857`.

24 Wei Dong. KGraph, 2016. URL: `http://www.kgraph.org/`.

25 Wei Dong, Moses Charikar, and Kai Li. Efficient k-nearest neighbor graph construction for generic similarity measures. In *WWW*, pages 577–586, 2011. `doi:10.1145/1963405.1963487`.

26 Moshe Dubiner. Bucketing coding and information theory for the statistical high-dimensional nearest-neighbor problem. *IEEE Transactions on Information Theory*, 56(8):4166–4179, Aug 2010. `doi:10.1109/TIT.2010.2050814`.

27 Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley, 2000.

28 David Eppstein, Michael S. Paterson, and F. Frances Yao. On nearest-neighbor graphs. *Discrete & Computational Geometry*, 17(3):263–282, 1997.

29 Kiana Hajebi, Yasin Abbasi-Yadkori, Hossein Shahbazi, and Hong Zhang. Fast approximate nearest-neighbor search with k-nearest neighbor graph. In *IJCAI*, volume 22, pages 1312–1317, 2011. doi:10.5591/978-1-57735-516-8/IJCAI11-222.

30 Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *STOC*, pages 604–613, 1998. doi:10.1145/276698.276876.

31 William B. Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics*, 26(1):189–206, 1984. doi:10.1090/conm/026/737400.

32 Christopher Kennedy and Rachel Ward. Fast cross-polytope locality-sensitive hashing. In *ITCS*, pages 53:1–53:16, 2017. doi:10.4230/LIPIcs.ITCS.2017.53.

33 Thijs Laarhoven. Sieving for shortest vectors in lattices using angular locality-sensitive hashing. In *CRYPTO*, pages 3–22, 2015. doi:10.1007/978-3-662-47989-6_1.

34 Thijs Laarhoven. Tradeoffs for nearest neighbors on the sphere. *arXiv*, pages 1–16, 2015. URL: https://arxiv.org/abs/1511.07527.

35 Thijs Laarhoven. Hypercube LSH for approximate near neighbors. In *MFCS*, pages 7:1–7:20, 2017. doi:10.4230/LIPIcs.MFCS.2017.7.

36 Y.A. Malkov and D.A. Yashunin. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. *arXiv:1603.09320*, pages 1–21, 2016. URL: https://arxiv.org/abs/1603.09320.

37 Yury Malkov, Alexander Ponomarenko, Andrey Logvinov, and Vladimir Krylov. Approximate nearest neighbor algorithm based on navigable small world graphs. *Information Systems*, 45:61–68, 2014. doi:10.1016/j.is.2013.10.006.

38 Alexander May and Ilya Ozerov. On computing nearest neighbors with applications to decoding of binary linear codes. In *EUROCRYPT*, pages 203–228, 2015. doi:10.1007/978-3-662-46800-5_9.

39 Gary L. Miller, Shang-Hua Teng, William Thurston, and Stephen A. Vavasis. Separators for sphere-packings and nearest neighbor graphs. *Journal of the ACM*, 44(1):1–29, 1997.

40 Marius Muja and David G. Lowe. FLANN, 2013. URL: https://www.cs.ubc.ca/research/flann/.

41 Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GloVe: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL: http://www.aclweb.org/anthology/D14-1162.

42 Erion Plaku and Lydia E. Kavraki. Distributed computation of the $k$nn graph for large high-dimensional point sets. *Journal of Parallel and Distributed Computing*, 67(3):346–359, 2007.

43 Alexander Ponomarenko, Yury Malkov, Andrey Logvinov, and Vladimir Krylov. Approximate nearest neighbor search small world approach. In *ICTA*, 2011. URL: http://www.iiis.org/CDs2011/CD2011IDI/ICTA_2011/Abstract.asp?myurl=CT175ON.pdf.

44 Ilya Razenshteyn and Ludwig Schmidt. FALCONN, 2016. URL: https://falconn-lib.org/.

45 Gregory Shakhnarovich, Trevor Darrell, and Piotr Indyk. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*. MIT Press, 2005.

46 Kengo Terasawa and Yuzuru Tanaka. Spherical LSH for approximate nearest neighbor search on unit hypersphere. In *WADS*, pages 27–38, 2007. doi:10.1007/978-3-540-73951-7_4.

47 Jing Wang, Jingdong Wang, Gang Zeng, Zhuowen Tu, Rui Gan, and Shipeng Li. Scalable k-nn graph construction for visual descriptors. In *CVPR*, pages 1106–1113, 2012. doi:10.1109/CVPR.2012.6247790.

A Nearly Optimal Algorithm for the Geodesic Voronoi Diagram of Points in a Simple Polygon

Chih-Hung Liu
Department of Computer Science, ETH Zürich, Zürich, Switzerland
chih-hung.liu@inf.ethz.ch
https://orcid.org/0000-0001-9683-5982

Abstract

The *geodesic Voronoi diagram* of m point sites inside a simple polygon of n vertices is a subdivision of the polygon into m cells, one to each site, such that all points in a cell share the same nearest site under the geodesic distance. The best known lower bound for the construction time is $\Omega(n+m\log m)$, and a matching upper bound is a long-standing open question. The state-of-the-art construction algorithms achieve $O\big((n+m)\log(n+m)\big)$ and $O(n+m\log m\log^2 n)$ time, which are optimal for $m=\Omega(n)$ and $m=O(\frac{n}{\log^3 n})$, respectively. In this paper, we give a construction algorithm with $O\big(n+m(\log m+\log^2 n)\big)$ time, and it is *nearly optimal* in the sense that if a single Voronoi vertex can be computed in $O(\log n)$ time, then the construction time will become the optimal $O(n+m\log m)$. In other words, we reduce the problem of constructing the diagram in the optimal time to the problem of computing a single Voronoi vertex in $O(\log n)$ time.

2012 ACM Subject Classification Theory of computation → Randomness, geometry and discrete structures

Keywords and phrases Simple polygons, Voronoi diagrams, Geodesic distance

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.58

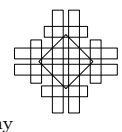
Related Version A full version of this paper is available at `http://arxiv.org/abs/1803.03526`.

1 Introduction

The *geodesic Voronoi diagram* of m point sites inside a simple polygon of n vertices is a subdivision of the polygon into m *cells*, one to each site, such that all points in a cell share the same *nearest site* where the distance between two points is the length of the shortest path between them inside the polygon. The common boundary between two cells is a *Voronoi edge*, and the endpoints of a Voronoi edge are *Voronoi vertices*. A cell can be augmented into *subcells* such that all points in a subcell share the same *anchor*, where the anchor of a point in the cell is the vertex of the shortest path from the associated site to the point that immediately precedes the point. An anchor is either a point site or a *reflex* polygon vertex. Figure 1(a) illustrates an augmented diagram.

The size of the (augmented) diagram is $\Theta(n+m)$ [1]. The best known construction time is $O\big((n+m)\log(n+m)\big)$ [10] and $O(n+m\log m\log^2 n)$ [9]. They are optimal for $m=\Omega(n)$ and for $m=O(\frac{n}{\log^3 n})$, respectively, since the best known lower bound is $\Omega(n+m\log m)$. The existence of a matching upper bound is a long-standing open question by Mitchell [8].

Aronov [1] first proved fundamental properties: a bisector between two sites is a simple curve consisting of $\Theta(n)$ straight and hyperbolic arcs and ending on the polygon boundary; the diagram has $\Theta(n+m)$ vertices, $\Theta(m)$ of which are Voronoi vertices. Then, he developed a divide-and-conquer algorithm that recursively partitions the polygon into two roughly equal-size sub-polygons. Since each recursion level takes $O\big((n+m)\log(n+m)\big)$ time to extend the diagrams between every pair of sub-polygons, the total time is $O\big((n+m)\log(n+m)\log n\big)$.




Papadopoulou and Lee [10] combined the divide-and-conquer and plane-sweep paradigms to improve the construction time to $O\big((n+m)\log(n+m)\big)$. First, the polygon is triangulated and one resultant triangle is selected as the root such that the dual graph is a rooted binary tree and each pair of adjacent triangles have a parent-child relation; see Figure 1(b). For each triangle, its diagonal shared with its parent partitions the polygon into two sub-polygons: the "*lower*" one contains it, and the "*upper*" one contains its parent. Then, the triangles are swept by the post-order and pre-order traversals of the rooted tree to respectively build, inside each triangle, the two diagrams with respect to sites in its lower and upper sub-polygons. Finally, the two diagrams inside each triangle are merged into the final diagram.

Very recently, Oh and Ahn [9] generalized the notion of plane sweep to a simple polygon. To supplant the scan line, one point is fixed on the polygon boundary, and another point is moved from the fixed point along the polygon boundary counterclockwise, so that the shortest path between the two points will sweep the whole polygon. Moreover, Guibas and Hershberger's data structure for shortest path queries [4, 6] is extended to compute a Voronoi vertex among three sites or between two sites in $O(\log^2 n)$ time. This technique enables handling an event in $O(\log m \log^2 n)$ time, leading to a total time of $O(n + m \log m \log^2 n)$.

Papadopoulou and Lee's method [10] has two issues inducing the $n\log(n+m)$ time-factor. First, while sweeping the polygon, an intermediate diagram is represented by a "*wavefront*" in which a "*wavelet*" is associated with a "subcell." Although this representation enables computing the common vertex among three subcells in $O(1)$ time, since it takes $\Omega\big(\log(n+m)\big)$ time to update such a wavefront, the $\Omega(n)$ vertices lead to the $n\log(n+m)$ factor. Second, when a wavefront enters a triangle from one diagonal and leaves from the other two diagonals, it will split into two. Since there are $\Omega(n)$ triangles, there are $\Omega(n)$ split events, and since a split event takes $\Omega\big(\log(n+m)\big)$ time, the $n\log(n+m)$ factor arises again.

1.1 Our contribution

We devise a construction algorithm with $O\big(n + m(\log m + \log^2 n)\big)$ time, which is slightly faster than Oh and Ahn's method [9] and is optimal for $m = O(\frac{n}{\log^2 n})$. More importantly, our algorithm is, to some extent, *nearly optimal* since the $\log^2 n$ factor solely comes from computing a single Voronoi vertex. If the computation time can be improved to $O(\log n)$, the total construction time will become $O\big(n + m(\log m + \log n)\big)$, which equals the optimal $O(n+m\log m)$ since $m\log n = O(n)$ for $m = O(\frac{n}{\log n})$ and $\log n = O(\log m)$ for $m = \Omega(\frac{n}{\log n})$. In other words, we reduce the problem of constructing the diagram in the optimal time by Mitchell [8] to the problem of computing a single Voronoi vertex in $O(\log n)$ time.

At a high level, our algorithm is a new implementation of Papadopoulou and Lee's concept [10] using a different data structure of a wavefront, symbolic maintenance of incomplete Voronoi edges, tailor-made wavefront operations, and appropriate amortized time analysis.

First, in our wavefront, each wavelet is directly associated with a cell rather than a subcell. This representation makes use of Oh and Ahn's [9] $O(\log^2 n)$-time technique of computing a Voronoi vertex. Each wavelet also stores the anchors of incomplete subcells in its associated cell in order to enable locating a point in a subcell along the wavefront.

Second, if each change of a wavefront needs to be updated immediately, a priority queue for events would be necessary, and since the diagram has $\Theta(m+n)$ vertices, an $(m+n)\log(m+n)$ time-factor would be inevitable. To overcome this issue, we maintain incomplete Voronoi edges symbolically, and update them only when necessary. For example, during a binary search along a wavefront, each incomplete Voronoi edge of a tested wavelet will be updated.

Third, to avoid $\Omega(n)$ split operations, we design two tailor-made operations. If a wavefront will separate into two but one part will not be involved in the follow-up "sweep", then, instead of using a binary search, we "divide" the wavefront in a seemingly brute-force way in which

we traverse the wavefront from the uninvolved part until the "division" point, remove all visited subcells, and build another wavefront from those subcells. If a wavefront propagates into a sub-polygon that contains no point site, then we adopt a two-phase process to build the diagram inside the sub-polygon instead of splitting a wavefront many times.

Finally, when deleting or inserting a subcell (anchor), its position in a wavelet is known. Since re-balancing a *red-black tree* (RB-tree) after an insertion or a deletion takes amortized $O(1)$ time [7, 11, 12], by augmenting each tree node with pointers to its predecessor and successor, an insertion or a deletion with a known position takes amortized $O(1)$ time.

This paper is organized as follows. Section 2 formulates the geodesic Voronoi diagram, defines a rooted partition tree, and introduces Papadopoulou and Lee's two subdivisions [10]; Section 3 summarizes our algorithm; Section 4 designs the data structure of a wavefront; Section 5 presents wavefront operations; Section 6 implements the algorithm with those operations. Due to the page limit, we omit some technical details and proofs; for more details, please see the full version.

2 Preliminary

2.1 Geodesic Voronoi diagrams

Let P be a simple polygon of n vertices, let ∂P denote the boundary of P, and let S be a set of m point sites inside P. For any two points p, q in P, the *geodesic distance* between them, denoted by $d(p, q)$, is the length of the shortest path between them that fully lies in P, the *anchor* of q with respect to p is the last vertex on the shortest path from p to q before q, and the *shortest path map* (SPM) from p in P is a subdivision of P such that all points in a region share the same anchor with respect to p. Each edge in the SPM from p is a line segment from a reflex polygon vertex v of P to ∂P along the direction from the anchor of v (with respect to p) to v, and this line segment is called the *SPM edge* of v (from p).

The *geodesic Voronoi diagram* of S in P, denoted by $\mathrm{Vor}_P(S)$, partitions P into m *cells*, one to each site, such that all points in a cell share the same nearest site in S under the geodesic distance. The cell of a site s can be augmented by partitioning the cell with the SPM from s into *subcells* such that all points in a subcell share the same anchor with respect to s. The augmented version of $\mathrm{Vor}_P(S)$ is denoted by $\mathrm{Vor}^*_P(S)$. With a slight abuse of terminology, a cell in $\mathrm{Vor}^*_P(S)$ indicates a cell in $\mathrm{Vor}_P(S)$ together with its subcells in $\mathrm{Vor}^*_P(S)$. Then, each cell is associated with a site, and each subcell is associated with an anchor. As shown in Figure 1(a), v is the anchor of the shaded subcell (in s_1's cell), and the last vertex on the shortest path from s_1 to any point x in the shaded subcell before x is v.

A *Voronoi edge* is the common boundary between two adjacent cells, and the endpoints of a Voronoi edge are called *Voronoi vertices*. A Voronoi edge is a part of the *geodesic bisector* between the two associated sites, and consists of straight and hyperbolic arcs. Endpoints of these arcs except Voronoi vertices are called *breakpoints*, and a breakpoint is incident to an SPM edge in the SPM from one of the two associated sites, indicating a change of the corresponding anchor. There are $\Theta(m)$ Voronoi vertices and $\Theta(n)$ breakpoints [1].

In our algorithm, each anchor u refers to either a reflex polygon vertex of P or a point site in S; we store its associated site s, its geodesic distance from s, and its anchor with respect to s. The weighted distance from u to a point x is $d(s, u) + |\overline{ux}|$.

Throughout the paper, we make a general position assumption that no polygon vertex is equidistant from two sites in S and no point inside P is equidistant from four sites in S. The former avoids nontrivial overlapping among cells [1], and the latter ensures that the degree of each Voronoi vertex with respect to Voronoi edges is either 1 (on ∂P) or 3 (among three cells).

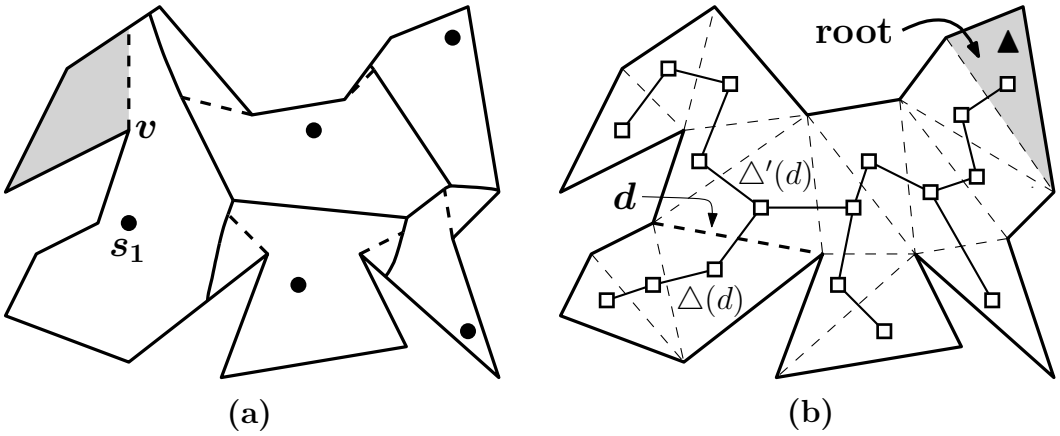


Figure 1 (a) Augmented geodesic Voronoi diagram $\mathrm{Vor}^*_P(S)$. (b) Rooted partition tree $\mathcal{T}$.

The boundary of a cell except Voronoi edges are polygonal chains on ∂P. For convenience, these polygonal chains are referred to as *polygonal edges* of the cell, the incidence of an SPM edge onto a polygonal edge is also a *breakpoint*, and the polygonal edges including their polygon vertices and breakpoints also count for the size of the cell.

▶ **Lemma 1.** *([9, Lemma 5 and 14]) It takes $O(\log^2 n)$ time to compute the degree-1 or degree-3 Voronoi vertex between two sites or among three sites after $O(n)$-time preprocessing.*

2.2 A rooted partition tree

Following Papadopoulou and Lee [10], a rooted partition tree $\mathcal{T}$ for P and S is built as follows: First, P is triangulated using Chazelle's algorithm [2] in $O(n)$ time, and all sites in S are located in the resulting triangles by Edelsbrunner et al's approach [3] in $O(n + m \log n)$ time. The dual graph of the triangulation is a tree in which each node corresponds to a triangle and an edge connects two nodes if and only if their corresponding triangles share a diagonal. Then, an arbitrary triangle ▲ whose two diagonals are polygon sides, i.e., a node with degree 1, is selected as the *root*, so that there is a parent-child relation between each pair of adjacent triangles. Figure 1(b) illustrates a rooted partition tree $\mathcal{T}$.

For a diagonal d, let $\triangle(d)$ and $\triangle'(d)$ be the two triangles adjacent to d such that $\triangle'(d)$ is the parent of $\triangle(d)$, and call d the *root diagonal* of $\triangle(d)$; also see Figure 1(b). d partitions P into two sub-polygons: $P(d)$ contains $\triangle(d)$ and $P'(d)$ contains $\triangle'(d)$. $P(d)$ and $P'(d)$ are said to be "below" and "above" d, respectively. Assume that $d \subseteq P(d)$; let $S(d) = S \cap P(d)$ and $S'(d) = S \setminus S(d)$, which indicate the two respective subsets of sites below and above d.

In this paper, we adopt the following convention: for each triangle $\triangle$, let $d = \overline{v_1 v_2}$ be its root diagonal, let $\triangle'$ be its parent triangle, let d_1, d_2 be the other two diagonals of $\triangle$, and let d_3, d_4 be the other two diagonals of $\triangle'$. Assume that d_4 is the diagonal between $\triangle'$ and its parent, and that d_1, d, and d_4 are incident to v_1, and d_2, d, and d_3 are incident to v_2. Let $v_{1,2}$ be the vertex shared by d_1 and d_2, and let $v_{3,4}$ be the vertex shared by d_3 and d_4. Denote the set of sites in $\triangle$ as $S_\triangle = S(d) - S(d_1) - S(d_2)$. Figure 2(a) shows an illustration.

2.3 Subdivisions

Papadopoulou and Lee [10] introduced two subdivisions, $\mathcal{SD}$ and $\mathcal{SD}'$, of P, which can be merged into $\mathrm{Vor}^*_P(S)$. For each triangle $\triangle$ with a root diagonal d, $\mathcal{SD}$ and $\mathcal{SD}'$ respectively contain $\mathrm{Vor}^*_P\big(S(d)\big) \cap \triangle$ and $\mathrm{Vor}^*_P\big(S'(d)\big) \cap \triangle$; $\mathcal{SD}$ also contains $\mathrm{Vor}^*_P(S) \cap$ ▲. Since $S(d)$

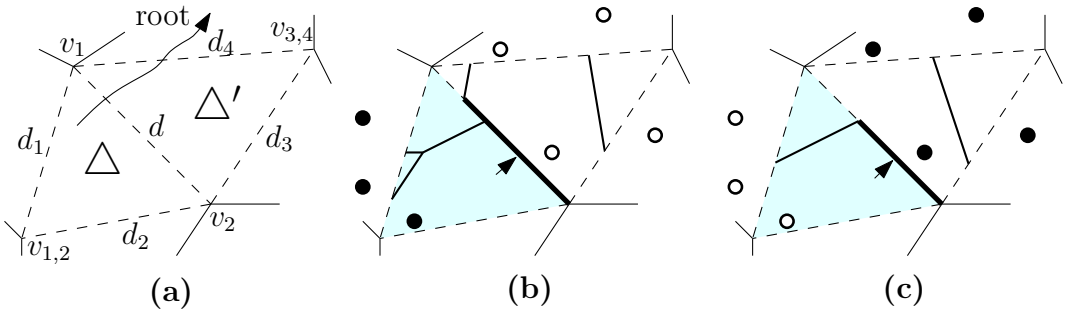


Figure 2 (a) $\triangle$ and $\triangle'$. (b) $\mathcal{SD} \cap \triangle$. (c) $\mathcal{SD}' \cap \triangle$. (Borders on d are indicated by arrows.)

and $S(d_4)$ (resp. $S'(d)$ and $S'(d_4)$) may differ, a *border* forms along d in $\mathcal{SD}$ (resp. $\mathcal{SD}'$) to "remedy" the conflicting proximity information. Figure 2(b)–(c) illustrate $\mathcal{SD}$ and $\mathcal{SD}'$.

The incidence of a Voronoi edge or an SPM edge in $\mathrm{Vor}_P^*(S)$ onto a border in $\mathcal{SD}$ or $\mathcal{SD}'$ is called a *border vertex*, and the border vertices partition a border into *border edges*. Both $\mathcal{SD}$ and $\mathcal{SD}'$ have $O(n+m)$ border vertices [10], $O(m)$ of which are induced by Voronoi edges. Hereafter, a *diagram vertex* means a Voronoi vertex, a breakpoint, a border vertex, or a polygon vertex.

3 Overview of the algorithm

We compute $\mathrm{Vor}_P^*(S)$ in the following three steps:

1. Build the rooted partition tree $\mathcal{T}$ for P and S in $O(n + m \log n)$ time. (Section 2.2)
2. Construct $\mathcal{SD}$ and $\mathcal{SD}'$ in $O\big(n + m(\log m + \log^2 n)\big)$ time by sweeping the polygon using the post-order and pre-order traversals of $\mathcal{T}$, respectively. (Section 6.1 and Section 6.2)
3. Merge $\mathcal{SD}$ and $\mathcal{SD}'$ into $\mathrm{Vor}_P^*(S)$ in $O(n+m)$ time using Papadopoulou and Lee's method [10, Section 7].

By the above-mentioned running times, we conclude the total running time as follows.

▶ **Theorem 2.** *$\mathrm{Vor}_P^*(S)$ can be constructed in $O\big(n + m(\log m + \log^2 n)\big)$ time.*

4 Wavefront structure

A *wavefront* represents the "*incomplete*" boundary of "*incomplete*" Voronoi cells during the execution of our algorithm, and wavefronts will "*sweep*" the simple polygon P triangle by triangle to construct $\mathcal{SD}$ and $\mathcal{SD}'$. To avoid excessive updates, each "*incomplete*" Voronoi edge, which is a part of a Voronoi edge and will be completed during the sweep, is maintained *symbolically*, preventing an extra $\log n$ time-factor. During the sweep, candidates for Voronoi vertices in $\mathcal{SD}$ and $\mathcal{SD}'$ called *potential vertices* will be generated in the *unswept* part of P.

4.1 Formal definition and data structure

Let η be a diagonal or a pair of diagonals sharing a common polygon vertex, and let S' be a subset of S lying on the same side of η. A *wavefront* $W_\eta(S')$ represents the sequence of Voronoi cells in $\mathrm{Vor}_P^*(S')$ appearing along η, and each appearance of a cell induces a *wavelet* in $W_\eta(S')$. The *unswept* area of $W_\eta(S')$ is the part of P on the opposite side of η from S'. Since $\mathrm{Vor}_P^*(S')$ in the unswept area has not yet been constructed, Voronoi and polygonal

edges incident to η are called *incomplete*. Each wavelet is bounded by two *incomplete* Voronoi or polygonal edges along η, and its *incomplete boundary* comprises its two incomplete edges and the portion of η between them. When the context is clear, a wavelet may indicate its associated cell.

$W_\eta(S')$ is stored in an RB-tree in which each node refers to one wavelet and the ordering of nodes follows their appearances along η. The RB-tree is augmented such that each node has pointers to its predecessor and successor, and the root has pointers to the first and last nodes, enabling efficiently traversing wavelets along η and accessing the two ending wavelets.

The subcells of a wavelet are the subcells in its associated cell incident to its incomplete boundary. The list of their anchors is also maintained by an augmented RB-tree in which their ordering follows their appearances along the incomplete boundary. Due to the visibility of a subcell, each subcell appears exactly once along the incomplete boundary. Since the rebalancing after an insertion or a deletion takes amortized $O(1)$ time [11, 12, 7], inserting or deleting an anchor at a known position in an RB-tree, i.e., without searching, takes amortized $O(1)$ time.

4.2 Incomplete Voronoi and polygonal edges

When a wavefront moves into its *unswept* area, incomplete Voronoi edges will extend, generating new breakpoints. If each breakpoint needs to be created immediately, all candidates for breakpoints should be maintained in a priority queue, leading to an $\Omega(n \log n)$ running time due to $\Omega(n)$ breakpoints. To avoid these excessive updates, we maintain each incomplete Voronoi edge *symbolically*, and update it only when necessary. For example, when searching along the wavefront or merging two wavefronts, the incomplete Voronoi edges of each involved wavelet (cell) will be updated until the diagonal or the pair of diagonals.

Since a breakpoint indicates the change of a corresponding anchor, for a Voronoi edge, if the anchors of its incident subcells on "each side" are stored in a sequence, the Voronoi edge can be computed in time proportional to the number of breakpoints by scanning the two sequences [9, Section 4]. Following this concept, for each incomplete Voronoi edge, we store its fixed Voronoi vertex (in the swept area), its last created breakpoint, and its last used anchors on its two sides, so that we can update an incomplete Voronoi edge by scanning the two lists of anchors from the last used anchors. When creating a breakpoint, we also build a corresponding SPM edge, and then remove the corresponding anchor from the anchor list.

Each polygonal edge is also maintained symbolically in a similar way; in particular, it will also be updated when a polygon vertex is inserted as an anchor. Meanwhile, the SPM edges incident to a polygonal edge will also be created using its corresponding anchor list.

4.3 Potential vertices

We process incomplete Voronoi edges to generate candidates for Voronoi vertices called *potential vertices*. For each incomplete Voronoi edge, since its two associated sites lie in the swept area, one endpoint of the corresponding bisector lies in the unswept area and is a degree-1 potential vertex. For each two adjacent Voronoi edges along the wavefront, their respective bisectors may intersect in the unswept area, and the intersection is a degree-3 potential vertex. By Lemma 1, a potential vertex can be computed in $O(\log^2 n)$ time.

Potential vertices are stored in their located triangles; each diagonal of a triangle is equipped with a priority queue to store the potential vertices associated with sites in the triangles on the opposite side of the diagonal, where the key is the distance to the diagonal.

5 Wavefront operations

We summarize the eight wavefront operations, where K_{inv}, A_{vis} and I_{new} are the numbers of involved (i.e., processed and newly created) potential vertices, visited anchors, and created diagram vertices, respectively:

Initiate: Compute $\text{Vor}_{\triangle}(S_{\triangle})$ and initialize $W_{(d,d_2)}(S_{\triangle})$ in $O\big(|S_{\triangle}|(\log m + \log^2 n)\big)$ time.

Extend: Extend one wavefront into a triangle from one diagonal to the opposite pair of diagonals to build the diagram inside the triangle in $O\big(K_{\text{inv}}(\log m + \log^2 n) + I_{\text{new}}\big)$ plus amortized $O(1)$ time.

Merge: Merge two wavefronts sharing the same diagonal or the same pair of diagonals together with merging the two corresponding diagrams in $O\big(|S_{\triangle}|(\log m + \log n) + K_{\text{inv}}(\log m + \log^2 n) + I_{\text{new}}\big)$ plus amortized $O(1)$ time where $\triangle$ is the underlying triangle.

Join: Join two wavefronts sharing the same diagonal with building the border on the diagonal but without merging the two corresponding diagrams inside the underlying triangle $\triangle'$ in $O\big(|S_{\triangle'}|(\log m + \log n) + K_{\text{inv}}(\log m + \log^2 n) + I_{\text{new}}\big)$ plus amortized $O(1)$ time.

Split: Split a wavefront using a binary search in $O(\log m + \log n)$ time.

Divide: Divide a wavefront by traversing one diagonal in amortized $O(A_{\text{vis}} + 1)$ time.

Insert: Insert $S_{\triangle}$ into $W_{d_1}\big(S(d_2)\big)$ to be $W_{d_1}\big(S(d_2) \cup S_{\triangle}\big)$ in $O\big(|S_{\triangle}|(\log m + \log^2 n)\big)$ time.

Propagate: Propagate $W_d\big(S'(d)\big)$ into $P(d)$, provided that $P(d) \cap S = S(d) = \emptyset$, to build $\mathcal{SD}' \cap P(d) = \text{Vor}^*_P(S) \cap P(d)$ in $O\big(K_{\text{inv}}(\log m + \log^2 n) + |\mathcal{SD}' \cap P(d)|\big)$ time.

Merge and Join operations differ in that the former also merges the two corresponding Voronoi diagrams inside the underlying triangle, and the latter does not.

Readers could directly read the algorithm in Section 6 without knowing the detailed implementation for wavefront operations. For the operation times, we have three remarks.

▶ Remark. During Extend and Propagate operations and at the end of the other operations, potential vertices will be generated according to new incomplete Voronoi edges. It will be clear in Section 6 that the total number of potential vertices in the construction of $\mathcal{SD}$ and $\mathcal{SD}'$ is $O(m)$, so a priority queue takes $O(\log m)$ time for an insertion or an extraction.

▶ Remark. As stated in Section 4.2, we maintain incomplete Voronoi/polygonal edges symbolically. For the sake of simplicity, we charge the time to update an incomplete edge to the created breakpoints, and assign the corresponding costs to their located triangles.

▶ Remark. Since a wavelet (resp. anchor) to remove from a wavefront (resp. wavelet) must be inserted beforehand, we charge its removal cost at its insertion time. For a wavelet, the cost is $O(\log m)$, and for an anchor, since the position is always known, the cost is amortized $O(1)$. Similarly, we charge the cost to delete a diagram vertex at its creation time.

Due to the page limit, we omit Initiate, Split, Insert, and Join operations. The first three are quite straightforward, and the last one is similar to a Merge operation; for more details, please see the full version.

5.1 Extend operation

An Extend operation extends a wavefront $W_{\tilde{d}}(Q)$ from one diagonal $\tilde{d}$ of a triangle $\tilde{\triangle}$ to the opposite pair of diagonals $(\tilde{d}_1, \tilde{d}_2)$ to construct $W_{(\tilde{d}_1,\tilde{d}_2)}(Q)$ and $\text{Vor}^*_P(Q) \cap \tilde{\triangle}$, where $\tilde{d} = \overline{\tilde{v}_1\tilde{v}_2}$, $\tilde{d}_1 = \overline{\tilde{v}_1\tilde{v}_{1,2}}$, and $\tilde{d}_2 = \overline{\tilde{v}_2\tilde{v}_{1,2}}$, and Q lies on the opposite side of $\tilde{d}$ from $\tilde{\triangle}$.

This operation is equivalent to sweeping the triangle with a scan line parallel to $\tilde{d}$ and processing each hit potential vertex. The next hit potential vertex is provided from the priority queue associated with $\tilde{d}$ (defined in Section 4.3), and will be processed in three phases: First, its validity is verified: for a degree-1 potential vertex, its incomplete Voronoi

edge should be alive in the wavefront, and for a degree-3 one, its two incomplete Voronoi edges should be still adjacent in the wavefront. Second, the one or two incomplete Voronoi edges are updated up to the potential vertex. Since a degree-1 potential vertex is incident to a polygonal edge, the polygonal edge is also updated.

Finally, when a potential vertex becomes a Voronoi vertex, a wavelet will be removed from the wavefront. For a degree-1 potential vertex, since the wavelet lies at one end of the wavefront, a polygonal edge is added to its adjacent wavelet; for a degree-3 potential vertex, since the removal makes two wavelets adjacent, a new incomplete Voronoi edge is created, and one degree-1 and two degree-3 potential vertices are computed accordingly.

After the extension, if $\tilde{v}_{1,2}$ is a reflex polygon vertex, the wavefront is further processed by three cases. If neither $\tilde{d}_1$ nor $\tilde{d}_2$ is a polygon side, $\tilde{v}_{1,2}$ will later be inserted as an anchor while dividing or splitting $W_{(\tilde{d}_1,\tilde{d}_2)}(Q)$ at $\tilde{v}_{1,2}$. If both $\tilde{d}_1$ and $\tilde{d}_2$ are polygon sides, all the subcells will be completed along $(\tilde{d}_1, \tilde{d}_2)$ and the wavefront will disappear. If only $\tilde{d}_1$ (resp. $\tilde{d}_2$) is a polygon side, all the subcells along $\tilde{d}_1$ (resp. $\tilde{d}_2$) excluding the one containing $\tilde{v}_{1,2}$ will be completed, and $\tilde{v}_{1,2}$ will be inserted into the corresponding wavelet as the last or first anchor.

The operation time is $O\big(K_{\text{inv}}(\log m + \log^2 n) + I_{\text{new}}\big)$ plus amortized $O(1)$, where K_{inv} is the number of involved (i.e., processed and newly created) potential vertices and I_{new} is the number of created diagram vertices. First, extracting a potential vertex from the priority queue takes $O(\log m)$ time, and verifying its validity takes $O(1)$ time. Second, updating incomplete Voronoi and polygonal edges takes time linear in the number of created breakpoints (Section 4.2), i.e., $O(I_{\text{new}})$ time in total. Third, since computing a potential vertex takes $O(\log^2 n)$ time, locating it takes $O(\log n)$ time, and inserting it into a priority queue takes $O(\log m)$ time, creating a new potential vertex takes $O(\log^2 n + \log m)$ time. Finally, completing subcells takes $O(I_{\text{new}})$ time, and inserting $\tilde{v}_{1,2}$ takes amortized $O(1)$ time since its position in the anchor list is known.

5.2 Merge operation

A Merge operation merges $W_\eta(Q)$ and $W_\eta(Q')$ into $W_\eta(Q \cup Q')$ together with $\mathrm{Vor}^*_P(Q) \cap \triangle$ and $\mathrm{Vor}^*_P(Q') \cap \triangle$ into $\mathrm{Vor}^*_P(Q \cup Q') \cap \triangle$ where $\triangle$ is the underlying triangle. In our algorithm, either $Q = S_\triangle$, $Q' = S(d_1)$, and $\eta = (d, d_2)$ or $Q = S_\triangle \cup S(d_1)$, $Q' = S(d_2)$, and $\eta = d$; in both cases, $S_\triangle \subseteq Q$. The border will form on $\partial\triangle \setminus \eta$, i.e., d_1 for the former case and (d_1, d_2) for the latter case. Although a wavefront only stores incomplete Voronoi cells, its associated diagram can still be accessed through the Voronoi edges of the stored cells. After the merge, new incomplete Voronoi edges will form, and their potential vertices will be created.

The Merge operation consists of two phases: (1) merge $\mathrm{Vor}^*_P(Q) \cap \triangle$ and $\mathrm{Vor}^*_P(Q') \cap \triangle$ into $\mathrm{Vor}^*_P(Q \cup Q') \cap \triangle$ and (2) merge $W_\eta(Q)$ and $W_\eta(Q')$ into $W_\eta(Q \cup Q')$.

The first phase is to construct so-called *merge curves*. A merge curve is a connected component consisting of border edges along $\partial\triangle \setminus \eta$ and Voronoi edges in $\mathrm{Vor}^*_P(Q \cup Q') \cap \triangle$ associated with one site in Q and one site in Q'; the ordering of merge curves is the ordering of their appearances along η. This phase is almost identical to the merge process by Papadopoulou and Lee [10, Section 5], but since our data structure for a wavefront is different from theirs, a binary search along a wavefront to find a starting endpoint for a merge curve requires a different implementation. Due to the page limit, we only state this difference here; for the details of tracing a merge curve, please see the full version.

Assume η to be oriented from v_1 to $v_{1,2}$ for $\eta = (d, d_2)$ and from v_1 to v_2 for $\eta = d$. By [10, Lemma 4–6], a merge curve called *initial* starts from $v_{1,2}$ for the latter case ($\eta = d$), but all other merge curves have both endpoints on η, and those endpoints are associated with one site in $S_\triangle$. Let $Q_\triangle$ be the set of sites in $S_\triangle$ that have a wavelet in $W_\eta(Q)$. If $\eta = (d, d_2)$, a

site in $Q_\triangle$ can have two wavelets in $W_\eta(Q)$, and with an abuse of terminology, such a site is imagined to have two copies, each to one wavelet. Since $Q_\triangle \subseteq Q$, finding a starting endpoint for each merge curve except the initial one is to test sites in $Q_\triangle$ following the ordering of their wavelets in $W_\eta(Q)$ along η. After finding a starting endpoint, the corresponding merge curve will be traced; when the tracing reaches η again, a stopping endpoint forms, and the first site in $Q_\triangle$ lying after the site inducing the stopping endpoint will be tested.

Let x be the next starting endpoint, which is unknown, and let s be the next site in $Q_\triangle$ to test. A two-level binary search on $W_\eta(Q')$ determines if s induces x, and if so, further determines the site $t \in Q'$ that induces x with s as well as the corresponding anchor.

The first-level binary search executes on the RB-tree for the wavelets in $W_\eta(Q')$, and each step determines for a site $q \in Q'$ if its cell lies before or after t's cell along η or if $q = t$. Let y_1 and y_2 denote the two intersection points between η and the two incomplete edges of s (in $W_\eta(Q)$), where y_1 lies before y_2 along η, and let z_1 and z_2 be the two points defined similarly for q (in $W_\eta(Q')$). Since s lies in $\triangle$, the distance between s and any point in η can be computed in $O(1)$ time. The two incomplete edges of q will be updated until z_1 and z_2, so that the distance from q to z_1 (resp. to z_2) can be computed from the corresponding anchor. For example, if u is the anchor of the subcell that contains z_1, $d(z_1, q) = |\overline{z_1 u}| + d(u, q)$.

The determination considers four cases. (Assume s and t induce the "starting" endpoint.)

- z_2 lies before y_1 (resp. z_1 lies after y_2): t's cell lies after (resp. before) q's cell.
- z_1 lies before y_1 and z_2 lies between y_1 and y_2 (resp. z_2 lies after y_2 and z_1 lies between y_1 and y_2): if z_2 is closer to q than to s (resp. z_1 is closer to q than to s), then t's cell lies after (resp. before) q's cell; otherwise, t is q.
- Both y_1 and y_2 lie between z_1 and z_2: t is q.
- Both z_1 and z_2 lie between y_1 and y_2: let x_s be the projection point of s onto η.
  - If x_s lies before z_1, then t's cell lies before q's cell.
  - If x_s lies after z_2: if z_2 is closer to q than to s, t's cell lies after q's cell; if both z_1 and z_2 are closer to s than to q, t's cell lies before q's cell; otherwise, $t = q$.
  - If x_s lies between z_1 and z_2: if z_1 is closer to s than to q, t's cell lies before q's cell; otherwise, $t = q$.

If the first-level search does not find t, then s does not induce the next starting endpoint x.

The second-level binary search executes on the RB-tree for t's anchor list to either determine the next starting endpoint x and t's corresponding anchor or report that s does not induce x. Let u be the current anchor of t to test, and let x_s be the projection point of s onto η. u's "interval" on η can be decided by checking u's two neighboring anchors. If u's interval lies after x_s, x lies before u's interval; otherwise, if both endpoints of u's interval are closer to (resp. farther from) t than to (resp. from) s, x lies after (resp. before) u's interval, and if one endpoint is closer to t than to s but the other is not, then x lies in u's interval and can be computed in $O(1)$ time since $d(x, s) = |\overline{xu}| + d(u, t)$. If the second-level binary search does not find such an interval, s does not induce the next starting endpoint x.

The second phase (i.e., merging $W_\eta(Q)$ and $W_\eta(Q')$ into $W_\eta(Q \cup Q')$) splits $W_\eta(Q)$ and $W_\eta(Q')$ at the endpoints of merge curves, and concatenates active parts at these endpoints where a part is called active if it contributes to $W_\eta(Q \cup Q')$. In fact, the active parts along η alternately come from $W_\eta(Q)$ and $W_\eta(Q')$. At each merging endpoint, the two cells become adjacent, generating a new incomplete Voronoi edge. Potential vertices of these incomplete Voronoi edges will be computed and inserted into the corresponding priority queues. For each ending polygon vertex of η, if it is reflex but has not yet been an anchor of $W_\eta(Q \cup Q')$, it will be inserted into its located wavelet as the first or the last anchor.

The total operation time is $O\big(|S_\triangle|(\log n + \log m) + K_{\text{new}}(\log m + \log^2 n) + I_{\text{new}}\big)$ plus amortized $O(1)$, where K_{new} is the number of created potential vertices and I_{new} is the number of created diagram vertices while merging the two diagrams. First, since each two-level binary search takes $O(\log m + \log n)$ time, finding starting points takes $O\big(|S_\triangle|(\log m + \log n)\big)$ time. Second, by [10, Section 5], tracing a merge curve takes time linear in the number of deleted and created vertices, but the time to delete vertices has been charged at their creation, implying that tracing all the merge curves takes $O(I_{\text{new}})$ time.

Third, an incomplete Voronoi edge generates at least one potential vertex, so the number of new incomplete Voronoi edges is $O(K_{\text{new}})$. Since an endpoint of a merge curve corresponds to a new incomplete Voronoi edge, there are $O(K_{\text{new}})$ split and $O(K_{\text{new}})$ concatenation operations, and since each operation takes $O(\log m + \log n)$ time, it takes $O\big(K_{\text{new}}(\log m + \log n)\big)$ time to merge the two wavefronts. By the same analysis in Section 5.1, creating K_{new} potential vertices takes $O\big(K_{\text{new}}(\log m + \log^2 n)\big)$ time. Finally, inserting an ending polygon vertex of η as an anchor takes amortized $O(1)$ time.

5.3 Divide operation

A Divide operation divides a wavefront associated with a pair of diagonals at the common polygon vertex by traversing one diagonal instead of using a binary search. Although a Divide operation seems a brute-force way compared to a Split operation, since a Split operation takes $\Omega(\log n + \log m)$ time and there are $\Omega(n)$ events to separate a wavefront, if only Split operations are adopted, the total construction time would be $\Omega\big(n(\log n + \log m)\big)$.

First, the wavefront is traversed from the end of the selected diagonal subcell by subcell, i.e., anchor by anchor, until reaching the common vertex. Then, the wavefront is separated at the common polygon vertex by removing all the visited anchors except the last one, duplicating the last one, and building a new wavefront for these "removed" anchors and the duplicate anchor from scratch. Finally, if the common polygon vertex is reflex, it is inserted into its located wavelets in both resultant wavefronts as an anchor without a binary search (since it is the first or last anchor of its located wavelets).

The total operation time is amortized $O(A_{\text{vis}} + 1)$, where A_{vis} is the number of visited anchors. Since each wavelet (resp. anchor) records its two neighboring wavelets (resp. anchors) in the augmented RB-tree, the time to locate the common polygon vertex is $O(A_{\text{vis}})$. Recall that a cell must have one subcell, and the time to remove a wavelet or an anchor has been charged when it was inserted. Finally, building the new wavefront from scratch takes amortized $O(A_{\text{vis}})$ time, and inserting the common vertex takes amortized $O(1)$ time.

5.4 Propagate operation

A Propagate operation propagates a wavefront $W_d\big(S'(d)\big)$ into $P(d)$, i.e., from the upside to the downside of d, to build $\mathcal{SD}' \cap P(d)$ provided that $S \cap P(d) = S(d) = \emptyset$. Since $S(d) = \emptyset$, then $\mathcal{SD}' \cap P(d)$ is exactly $\text{Vor}_P^*(S) \cap P(d)$. To some extent, a Propagate operation is a generalized version of an Extend operation underlying a sub-polygon instead of a triangle.

The operation consists of two phases. The first phase constructs $\text{Vor}_P(S) \cap P(d)$, and the second phase refines each cell into subcells to obtain $\text{Vor}_P^*(S) \cap P(d)$.

The first phase "sweeps" the triangles in $P(d)$ by a preorder traversal of the subtree of $\mathcal{T}$ rooted at $\triangle(d)$. Similar to the Extend operation, this sweep processes potential vertices inside each triangle to construct Voronoi edges and to update the wavefront accordingly. However, this sweep will not process the polygon vertices of $P(d)$, so that the anchor lists will not be updated, preventing constructing a Voronoi edge using the two corresponding

anchor lists. Fortunately, Oh and Ahn [9] gave another technique that obtains in $O(\log n)$ time the two anchor lists of a Voronoi edge provided that the two Voronoi vertices are given, so a Voronoi edge can still be built in time proportional to $\log n$ plus the number of its breakpoints.

Therefore, the first phase takes $O\big(K_d(\log m + \log^2 n) + |\mathrm{Vor}_P(S) \cap P(d)|\big)$ time, where K_d is the number of involved potential vertices. Note that for the Voronoi edges that intersect d, their breakpoints outside $P(d)$ will be counted in the respective triangles in $P'(d)$.

The second phase triangulates each cell in $\mathrm{Vor}_P(S) \cap P(d)$ and constructs the SPM from the associated site in each triangulated cell. Since Chazelle's algorithm [2] takes linear time to triangulate a simple polygon and Guibas et al's algorithm [5] takes linear time to build the SPM in a triangulated polygon, the second phase takes $O(|\mathcal{SD}' \cap P(d)|)$ time.

▶ Remark. Although all the sites lie outside $P(d)$, the information stored in $W_d\big(S'(d)\big)$ allows us not to conduct Guibas et al's algorithm from scratch. For example, for each anchor u, its anchor $a(u)$ is also stored, and the SPM edge of u is the line segment from u to the polygon boundary along the direction from $a(u)$ to u.

To sum up, a Propagate operation takes $O\big(K_d(\log m + \log^2 n) + |\mathcal{SD}' \cap P(d)|\big)$ time.

6 Subdivision construction

6.1 Construction of $\mathcal{SD}$

To construct $\mathcal{SD}$, we process each triangle $\triangle$ by the postorder traversal of the rooted partition tree $\mathcal{T}$ and build $\mathcal{SD} \cap \triangle$. We first assume that no diagonal of $\triangle$ is a polygon side, and we will discuss the other cases later. Let d be the root diagonal of $\triangle$ and adopt the convention in Section 2.2 and Section 2.3. When processing $\triangle$, since its two children, $\triangle(d_1)$ and $\triangle(d_2)$, have been processed, $W_{d_1}\big(S(d_1)\big)$ and $W_{d_2}\big(S(d_2)\big)$ are available.

The processing of each triangle $\triangle$ consists of 8 steps:

1. Initiate $\mathrm{Vor}_\triangle(S_\triangle)$ and $W_{(d,d_2)}(S_\triangle)$.
2. Extend $W_{d_1}\big(S(d_1)\big)$ into $\triangle$ to generate $W_{(d,d_2)}\big(S(d_1)\big)$ and construct $\mathrm{Vor}^*_P\big(S(d_1)\big) \cap \triangle$.
3. Merge $W_{(d,d_2)}(S_\triangle)$ and $W_{(d,d_2)}\big(S(d_1)\big)$ into $W_{(d,d_2)}\big(S(d_1) \cup S_\triangle\big)$ by which $\mathrm{Vor}_\triangle(S_\triangle)$ and $\mathrm{Vor}^*_P\big(S(d_1)\big) \cap \triangle$ are merged into $\mathrm{Vor}^*_P\big(S(d_1) \cup S_\triangle\big) \cap \triangle$.
4. Divide $W_{(d,d_2)}\big(S(d_1) \cup S_\triangle\big)$ into $W_d\big(S(d_1) \cup S_\triangle\big)$ and $W_{d_2}\big(S(d_1) \cup S_\triangle\big)$ along d_2.
5. Extend $W_{d_2}\big(S(d_2)\big)$ into $\triangle$ to generate $W_{(d_1,d)}\big(S(d_2)\big)$ and construct $\mathrm{Vor}^*_P\big(S(d_2)\big) \cap \triangle$.
6. Divide $W_{(d_1,d)}\big(S(d_2)\big)$ into $W_{d_1}\big(S(d_2)\big)$ and $W_d\big(S(d_2)\big)$ along d_1.
7. Insert $S_\triangle$ into $W_{d_1}\big(S(d_2)\big)$ to obtain $W_{d_1}\big(S(d_2) \cup S_\triangle\big)$.
8. Merge $W_d\big(S(d_1) \cup S_\triangle\big)$ and $W_d\big(S(d_2)\big)$ into $W_d\big(S(d)\big)$ by which $\mathrm{Vor}^*_P\big(S(d_1) \cup S_\triangle\big) \cap \triangle$ and $\mathrm{Vor}^*_P\big(S(d_2)\big) \cap \triangle$ are merged into $\mathrm{Vor}^*_P\big(S(d)\big) \cap \triangle = \mathcal{SD} \cap \triangle$.

We remark that $W_{d_1}\big(S(d_2) \cup S_\triangle\big)$ and $W_{d_2}\big(S(d_1) \cup S_\triangle\big)$ will be used to construct $\mathcal{SD}'$.

If exactly one diagonal d of $\triangle$ is not a polygon side, it is either the root triangle $\blacktriangle$ or a leaf triangle. For the former, compute $\mathrm{Vor}_\blacktriangle(S_\blacktriangle)$, extend $W_d(S(d))$ into $\blacktriangle$ to build $\mathrm{Vor}^*_P(S(d)) \cap \blacktriangle$, and merge $\mathrm{Vor}_\blacktriangle(S_\blacktriangle)$ and $\mathrm{Vor}^*_P(S(d)) \cap \blacktriangle$ into $\mathrm{Vor}^*_P(S) \cap \blacktriangle = \mathcal{SD} \cap \blacktriangle$; for the latter, compute $\mathrm{Vor}_\triangle(S_\triangle)$ and initiate $W_d(S_\triangle)$. If exactly two diagonals, d and d', of $\triangle$ are not polygon sides, where d is the root diagonal, then compute $\mathrm{Vor}_\triangle(S_\triangle)$ to initiate $W_d(S_\triangle)$ and $W_{d'}(S_\triangle)$, extend $W_{d'}\big(S(d')\big)$ into $\triangle$ to obtain $W_d\big(S(d')\big)$ and $\mathrm{Vor}^*_P\big(S(d')\big) \cap \triangle$, and merge $W_d(S_\triangle)$ and $W_d\big(S(d')\big)$ to obtain $W_d\big(S(d)\big)$ and $\mathrm{Vor}^*_P\big(S(d)\big) \cap \triangle = \mathcal{SD} \cap \triangle$.

By the operation times in Section 5, the time to process $\triangle$ is summarized as follows.

▶ **Lemma 3.** *It takes $O\big((|S_\triangle| + K_\triangle)(\log m + \log^2 n) + I_\triangle\big)$ plus amortized $O(A_\triangle + 1)$ time to process $\triangle$, where $K_\triangle$ is the number of involved potential vertices, $I_\triangle$ is the number of created diagram vertices, and $A_\triangle$ is the number of visited anchors in Steps 4 and 6.*

To apply Lemma 3, we bound $\sum_\triangle K_\triangle$, $\sum_\triangle I_\triangle$ and $\sum_\triangle A_\triangle$ by the following two lemmas.

▶ **Lemma 4.** *In the construction of $\mathcal{SD}$, $\sum_\triangle K_\triangle = O(m)$ and $\sum_\triangle I_\triangle = O(n + m)$.*

Proof. There are 6 intermediate diagrams: $\mathrm{Vor}_\triangle(S_\triangle)$ (step 1), $\mathrm{Vor}^*_P\big(S(d_1)\big) \cap \triangle$ (step 2), $\mathrm{Vor}^*_P\big(S(d_1) \cup S_\triangle\big) \cap \triangle$ (step 3), $\mathrm{Vor}^*_P\big(S(d_2)\big) \cap \triangle$ (step 5), $\mathrm{Vor}^*_P\big(S(d_2) \cup S_\triangle\big) \cap \triangle$ (step 7), and $\mathrm{Vor}^*_P\big(S(d)\big) \cap \triangle = \mathcal{SD} \cap \triangle$ (step 8). First, a potential vertex arises due to the formation of an incomplete Voronoi edge, and an incomplete Voronoi edge generates $O(1)$ potential vertices. For the first diagram, the total number of Voronoi edges is $O(\sum_\triangle |S_\triangle|) = O(|S|) = O(m)$. For each of the other 5 diagrams, we can define borders in a similar way to $\mathcal{SD}$ and thus obtain a subdivision of P. Since each site has at most one cell in each resultant subdivision, Euler's formula implies that each subdivision contains $O(m)$ Voronoi edges among cells, leading to the conclusion that $\sum_\triangle K_\triangle = O(m)$. Second, a created diagram vertex must be a vertex of the first diagram or the other 5 subdivisions. Since there are m sites, the first diagram results in $O(m)$ vertices for all the triangles, and by the same reasoning of [10, Lemma 3], each subdivision has $O(n + m)$ vertice, leading to that $\sum_\triangle I_\triangle = O(n + m)$. ◀

▶ **Lemma 5.** *In the construction of $\mathcal{SD}$, $\sum_\triangle A_\triangle = O(n + m)$.*

Proof. We consider Step 4 (divide $W_{(d,d_2)}\big(S(d_1) \cup S_\triangle\big)$ along d_2), which is similar to Step 6. Since there are $O(n + m)$ anchors, it is sufficient to bound the number of anchors that are visited by Step 4 but still involved in the future construction of $\mathcal{SD}$, namely $\mathcal{SD} \cap P'(d)$. Since the subcell of each visited anchor intersects d_2, if the subcell does not intersect d, its anchor will not be involved in constructing $\mathcal{SD} \cap P'(d)$. A subcell of a visited anchor intersects d in two cases. In the first case, the subcell contains v_2 and thus intersects both d and d_2. Since there are $O(n)$ triangles, the total number for the first case is $O(n)$. In the second case, the subcell intersects both d and d_2 but does not contain v_2. By the definition of $W_{(d,d_2)}\big(S(d_1) \cup S_\triangle\big)$, its associated "site" belongs to either $S_\triangle$ or $S(d_1)$. For the former, since its anchor must be the site itself, the total number is $\sum_\triangle S_\triangle = m$. For the latter, since all the sites in $S(d_1)$ lie outside $\triangle$, only one site in $S(d_1)$ can own a cell intersecting both d and d_2. Moreover, due to the visibility of a subcell, only one subcell in such a cell can intersect both d and d_2. Since there are $O(n)$ triangles, the total number is $O(n)$. ◀

By Lemma 3, 4, and 5, we conclude the construction time of $\mathcal{SD}$ as follows.

▶ **Theorem 6.** *$\mathcal{SD}$ can be constructed in $O\big(n + m(\log m + \log^2 n)\big)$ time.*

Proof. By Lemma 3, we need to bound $\sum_\triangle \big((|S_\triangle| + K_\triangle) \cdot (\log m + \log^2 n) + I_\triangle + A_\triangle + 1\big)$. It is trivial that $\sum_\triangle |S_\triangle| = |S| = m$ and $\sum_\triangle 1 = O(n)$. By Lemma 4 and Lemma 5, $\sum_\triangle K_\triangle = O(m)$, $\sum_\triangle I_\triangle = O(n+m)$, and $\sum_\triangle A_\triangle = O(n+m)$, leading to the statement. ◀

6.2 Construction of $\mathcal{SD}'$

To construct $\mathcal{SD}'$, we processes each triangle by the preorder traversal of the partition tree $\mathcal{T}$. For the root triangle $\blacktriangle$, let $\tilde{d}$ be its diagonal that is not a polygon side, build $W_{\tilde{d}}(S_\blacktriangle) = W_{\tilde{d}}\big(S'(\tilde{d})\big)$, and if $S_\blacktriangle = S$, further propagate $W_{\tilde{d}}(S_\blacktriangle)$ into $P(\tilde{d})$. For other triangles $\triangle$, we assume that neither $\triangle$ nor its parent $\triangle'$ has a polygon side; the other cases can be processed in a similar way. Since $\triangle'$ has been processed, $W_d\big(S'(d)\big)$ is available, and by the construction of $\mathcal{SD}$, $W_{d_2}\big(S(d_1) \cup S_\triangle\big)$ and $W_{d_1}\big(S(d_2) \cup S_\triangle\big)$ have been generated.

If $S(d) \neq \emptyset$, $\triangle$ is processed by the following 4 steps:

1. Extend $W_d\big(S'(d)\big)$ into $\triangle$ to obtain $W_{(d_1,d_2)}(S'(d))$ and $\mathrm{Vor}_P^*\big(S'(d)\big) \cap \triangle = \mathcal{SD}' \cap \triangle$.
2. If neither $S(d_1)$ nor $S(d_2)$ is empty, split $W_{(d_1,d_2)}(S'(d))$ into $W_{d_1}(S'(d))$ and $W_{d_2}(S'(d))$; otherwise, if $S(d_1)$ (resp. $S(d_2)$) is empty, divide $W_{(d_1,d_2)}(S'(d))$ into $W_{d_1}(S'(d))$ and $W_{d_2}(S'(d))$ along d_1 (resp. d_2).
3. Join $W_{d_1}\big(S(d_2) \cup S_\triangle\big)$ and $W_{d_1}(S'(d))$ into $W_{d_1}(S'(d_1))$, and join $W_{d_2}\big(S(d_1) \cup S_\triangle\big)$ and $W_{d_2}(S'(d))$ into $W_{d_2}(S'(d_2))$.
4. If $S(d_1)$ (resp. $S(d_2)$) is empty, propagate $W_{d_1}(S'(d_1))$ (resp. $W_{d_2}(S'(d_2))$) into $P(d_1)$ (resp. $P(d_2)$) to build $\mathcal{SD}' \cap P(d_1)$ (resp. $\mathcal{SD}' \cap P(d_2)$).

We first analyze Split and Propagate operations.

▶ **Lemma 7.** *The total number of Split operations is $O(m)$.*

Proof. A Split operation occurs only if neither $S(d_1)$ nor $S(d_2)$ is empty. We recursively remove leaf nodes (triangles) containing no site from $\mathcal{T}$, so that each leaf node in the resulting tree $\mathcal{T}'$ contains a site. Then a Split operation occurs in an internal node of $\mathcal{T}'$ with two children. Since there are m sites, $\mathcal{T}'$ has $O(m)$ leaf nodes, and since $\mathcal{T}'$ is a binary tree, the number of internal nodes with two children is $O(m)$, leading to $O(m)$ Split operations. ◀

▶ **Lemma 8.** *The total time for all the Propagate operations is $O\big(n + m(\log m + \log^2 n)\big)$.*

Proof. For a sub-polygon to propagate, since each edge of the resultant subdivision has a vertex in the sub-polygon, the number of created edges is bounded by the number of created vertices. Therefore, by Section 5.4, the total time is $O\big(K(\log m + \log^2 n) + I\big)$, where K is the number of involved potential vertices and I is the number of created diagram vertices. Moreover, by the same reasoning of Lemma 4, $K = O(m)$, and since all the sub-polygons to propagate are disjoint, $I = O(|\mathcal{SD}'|) = O(m + n)$, leading to the statement. ◀

By Lemma 7, Lemma 8 and the reasoning of Theorem 6, we conclude the construction time of $\mathcal{SD}'$ as follows.

▶ **Theorem 9.** *$\mathcal{SD}'$ can be constructed in $O\big(n + m(\log m + \log^2 n)\big)$ time.*

References

1. Boris Aronov. On the geodesic Voronoi diagram of point sites in a simple polygon. *Algorithmica*, 4(1-4):109–140, 1989.
2. Bernard Chazelle. Triangulating a simple polygon in linear time. *Discrete & Computational Geometry*, 6:485–524, 1991.
3. Herbert Edelsbrunner, Leonidas J. Guibas, and Jorge Stolfi. Optimal point location in a monotone subdivision. *SIAM Journal on Computing*, 15(2):317–340, 1986.
4. 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.
5. Leonidas J. Guibas, John Hershberger, Daniel Leven, Micha Sharir, and Robert E. Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2(1-4):209–233, 1987.
6. John Hershberger. A new data structure for shortest path queries in a simple polygon. *Information Processing Letters*, 38(5):231–235, 1991.
7. Kurt Mehlhorn and Peter Sanders. Sorted sequences. In *Algorithms and Data Structures: The Basic Toolbox*. Springer-Verlag Berlin Heidelberg, 2008.
8. Joseph S. B. Mitchell. Geometric shortest paths and network optimization. In *Handbook of Computational Geometry*, pages 633–701. Elsevier, 2000.

9 Eunjin Oh and Hee-Kap Ahn. Voronoi diagrams for a moderate-sized point-set in a simple polygon. In *33rd International Symposium on Computational Geometry, SoCG 2017, July 4-7, 2017, Brisbane, Australia*, pages 52:1–52:15, 2017.

10 Evanthia Papadopoulou and D. T. Lee. A new approach for the geodesic Voronoi diagram of points in a simple polygon and other restricted polygonal domains. *Algorithmica*, 20(4):319–352, 1998.

11 Robert Endre Tarjan. Updating a balanced search tree in O(1) rotations. *Information Processing Letters*, 16(5):253–257, 1983.

12 Robert Endre Tarjan. Efficient Top-Down Updating of Red-Black Trees. Technical report, Technical Report TR-006-85. Dapartment of Computer Science, Princeton University, 1985.

Further Consequences of the Colorful Helly Hypothesis

Leonardo Martínez-Sandoval[1]
Department of Computer Science, Ben-Gurion University of the Negev
Beer-Sheva, Israel 84105
leomtz@im.unam.mx
https://orcid.org/0000-0002-5104-9635

Edgardo Roldán-Pensado[2]
Centro de Ciencias Matemáticas
UNAM campus Morelia, Michoacán, México 58190
e.roldan@im.unam.mx
https://orcid.org/0000-0002-2164-066X

Natan Rubin[3]
Department of Computer Science, Ben-Gurion University of the Negev
Beer-Sheva, Israel 84105
rubinnat.ac@gmail.com

Abstract

Let $\mathcal{F}$ be a family of convex sets in $\mathbb{R}^d$, which are colored with $d+1$ colors. We say that $\mathcal{F}$ satisfies the Colorful Helly Property if every rainbow selection of $d+1$ sets, one set from each color class, has a non-empty common intersection. The Colorful Helly Theorem of Lovász states that for any such colorful family $\mathcal{F}$ there is a color class $\mathcal{F}_i \subset \mathcal{F}$, for $1 \leq i \leq d+1$, whose sets have a non-empty intersection. We establish further consequences of the Colorful Helly hypothesis. In particular, we show that for each dimension $d \geq 2$ there exist numbers $f(d)$ and $g(d)$ with the following property: either one can find an additional color class whose sets can be pierced by $f(d)$ points, or all the sets in $\mathcal{F}$ can be crossed by $g(d)$ lines.

2012 ACM Subject Classification Mathematics of computing → Combinatorics, Theory of computation → Computational geometry

Keywords and phrases geometric transversals, convex sets, colorful Helly-type theorems, line transversals, weak epsilon-nets, transversal numbers

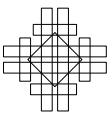
Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.59

Related Version A full version of this paper is available at [17], `https://arxiv.org/abs/1803.06229`.

Funding The project leading to this application has received funding from European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No. 678765.

Acknowledgements The authors thank the anonymous SoCG referees for valuable comments which helped to improve the presentation of the paper.

[1] Also supported by grant 1452/15 from Israel Science Foundation
[2] Also supported by PAPIIT project IA102118
[3] Also supported by grant 1452/15 from Israel Science Foundation, by Ralph Selig Career Development Chair in Information Theory and grant 2014384 from the U.S.-Israeli Binational Science Foundation



1 Introduction

1.1 Helly-type theorems

Let $\mathcal{F}$ be a finite family of convex sets in $\mathbb{R}^d$. We say that a collection X of geometric objects (e.g., points, lines, or k-flats – k-dimensional affine subspaces of $\mathbb{R}^d$) is a *transversal* to $\mathcal{F}$, or that $\mathcal{F}$ can be *pierced* or *crossed* by X, if each set of $\mathcal{F}$ is intersected by some member of X. For an integer j we use the symbol $\binom{\mathcal{F}}{j}$ to denote the collection of subfamilies of $\mathcal{F}$ of size j.

The 1913 theorem of Helly [14] states that a finite family $\mathcal{F}$ of convex sets has a non-empty intersection (i.e., $\mathcal{F}$ can be pierced by a single point) if and only if each of its subsets $\mathcal{F}' \subset \mathcal{F}$ of size at most $d+1$ can be pierced by a point.

In the past 50 years Geometric Transversal Theory has been preoccupied with the following questions (see e.g. [5], [10], [11], [12], [19]):

- Does Helly's Theorem generalize to transversals by k-flats, for $1 \leq k \leq d-1$?
- Given that a significant fraction of the $(d+1)$-tuples $\mathcal{F}' \in \binom{\mathcal{F}}{d+1}$ have a non-empty intersection, can $\mathcal{F}$, or at least some fixed fraction of its members, be pierced by constantly many points?

The first question has been settled to the negative already for $k = 1$. For instance, Santaló [18] and Danzer [9] observed that for any $n \geq 3$ there are families $\mathcal{F}$ of n convex sets in $\mathbb{R}^2$ so that any $n-1$ of the sets can be crossed by a single line transversal while no such transversal exists for $\mathcal{F}$. Nevertheless, Alon and Kalai [2] show that the following almost-Helly property holds for $k = d-1$: If every $d+1$ (or fewer) of the sets of $\mathcal{F}$ can be crossed by a hyperplane, then $\mathcal{F}$ admits a transversal by h hyperplanes, where the number $h = h(d)$ depends only on the dimension d.

While the properties of hyperplane transversals largely resemble those of point transversals, this is not the case for transversals by k-flats of intermediate dimensions $1 \leq k \leq d-2$. For example, Alon et al. [3] showed that for every integers $d \geq 3, m$ and $n_0 \geq m+4$ there is a family of at least n_0 convex sets so that any m of the sets can be crossed by a line but no $m+4$ of them can; this phenomenon can be largely attributed to the complex topological structure of the space of transversal k-flats.

The second question gave rise to a plethora of inter-related results in discrete geometry and topological combinatorics.

▶ **Theorem 1** (Fractional Helly's Theorem). *For any $d \geq 1$ and $\alpha > 0$ there is a number $\beta = \beta(\alpha, d) > 0$ with the following property: For every finite family $\mathcal{F}$ of convex sets in $\mathbb{R}^d$ so that at least $\alpha\binom{|\mathcal{F}|}{d+1}$ of the $(d+1)$-subsets $\mathcal{F}' \in \binom{\mathcal{F}}{d+1}$ have non-empty intersection, there is a point which pierces at least $\beta|\mathcal{F}|$ of the sets of $\mathcal{F}$.*

Theorem 1 was proved by Liu and Katchalski in [16] and it is one of the key ingredients in the proof of the so called Hadwiger-Debrunner (p, q)-Conjecture [13] by Alon and Kleitman [4].

▶ **Definition 2.** We say that a family of convex sets has the (p, q)-*property*, for $p \geq q$, if for any p-subset $\mathcal{F}' \in \binom{\mathcal{F}}{p}$ there is a q-subset $\mathcal{F}'' \in \binom{\mathcal{F}'}{q}$ with non-empty common intersection $\bigcap \mathcal{F}'' \neq \emptyset$.

▶ **Theorem 3** (The (p, q)-theorem [4]). *For any $d \geq 1$ and $p \geq q \geq d+1$ there is a number $P = P(p, q, d)$ with the following property: Any finite family $\mathcal{F}$ of convex sets in $\mathbb{R}^d$ with the (p, q)-property can be pierced by P points.*

The proof of Theorem 3 combines Theorem 1 with the following result of independent interest.

▶ **Theorem 4** (Weak ϵ-net for points [1]). *For any dimension $d \geq 1$ and $\epsilon > 0$ there is $W = W(\epsilon, d)$ with the following property: For every finite (multi-)set P of points in $\mathbb{R}^d$ one can find W points in $\mathbb{R}^d$ that pierce every convex set $A \subseteq \mathbb{R}^d$ with $|A \cap P| \geq \epsilon |P|$.*

Understanding the asymptotic behaviour of $W(\epsilon, d)$ is one of the most challenging open problems in discrete geometry.

The starting point of our investigation is the Colorful Helly Theorem of László Lovász, first stated in [7], which concerns the scenario in which the intersecting $(d+1)$-tuples form a complete $(d+1)$-partite hypergraph.

▶ **Definition 5.** We say that a finite family of convex sets $\mathcal{F}$ is *k-colored* if each set $K \in \mathcal{F}$ is colored with (at least) one of k distinct colors. The k-coloring of $\mathcal{F}$ can be expressed by writing $\mathcal{F}$ as a union of k *color classes* $\mathcal{F}_1 \cup \mathcal{F}_2 \cup \cdots \cup \mathcal{F}_k$, where each class $\mathcal{F}_i$ consists of the sets with color $i \in [k]$. We say that the k-colored family $\mathcal{F}$, with color classes $\mathcal{F}_1, \ldots, \mathcal{F}_k$, has the *Colorful Helly property, or* $\mathcal{CH}(\mathcal{F}_1, \ldots, \mathcal{F}_k)$ if every rainbow selection $K_i \in \mathcal{F}_i$, for $1 \leq i \leq k$, has non-empty intersection $\bigcap_{i=1}^{k} K_i \neq \emptyset$.

▶ **Theorem 6** (Colorful Helly's Theorem). *Let $\mathcal{F}$ be a $(d+1)$-colored family of convex sets in $\mathbb{R}^d$, with color classes $\mathcal{F}_1, \ldots, \mathcal{F}_{d+1}$. Then $\mathcal{CH}(\mathcal{F}_1, \ldots, \mathcal{F}_{d+1})$ implies that there is a color class $\mathcal{F}_i$ with non-empty intersection $\bigcap \mathcal{F}_i \neq \emptyset$.*

Notice that Theorem 6 says nothing about transversals to the remaining d color classes $\mathcal{F}_j$, with $j \in [d+1] \setminus \{i\}$. The primary goal of this paper is to gain a deeper understanding of the transversals to *all* of the color classes $\mathcal{F}_i$ in a $(d+1)$-colored family $\mathcal{F}$ that satisfies $\mathcal{CH}$.

Theorem 6 is in close relation, via point-hyperplane duality, with the colorful version of the Carathéodory theorem due to Bárány [7]. Holmsen *et al.* [15] and independently Arocha *et al.* [6] recently established the following strengthening of Bárány's result:

▶ **Theorem 7** (Very Colorful Carathéodory Theorem). *Let P be a finite set of points in $\mathbb{R}^d$ colored with $d+1$ colors. If every $(d+1)$-colorful subset of P is separated from the origin, then there exist two colors such that the subset of all points of these colors is separated from the origin.*

Unfortunately, there is no Very Colorful Helly Theorem which guarantees that a second color class can be pierced with few points, as is illustrated by the following example (see Figure 1). Let $\mathcal{F}_{d+1} = \{\mathbb{R}^d\}$ and, for each $1 \leq i \leq d$ let $\mathcal{F}_i$ be a collection of hyperplanes orthogonal to the x_i-axis. Then $\mathcal{F}_{d+1}$ is the only class that has a point transversal, moreover, each of the remaining classes may need an arbitrarily large number of points in order to be pierced. Note, though, that one can cross all the sets of $\bigcup_{i=1}^{d+1} \mathcal{F}_i$ by a *single line*.

1.2 Our results

Our main result suggests that, in a sense, the scenario in Figure 1 is the only possible unless an additional color class can be pierced by few points.

▶ **Theorem 8.** *For each dimension $d \geq 2$ there exist numbers $f(d)$ and $g(d)$ with the following property. Let $\mathcal{F}$ be a finite $(d+1)$-colored family of convex sets in $\mathbb{R}^d$ (with color classes $\mathcal{F}_1, \ldots, \mathcal{F}_{d+1}$) that satisfies $\mathcal{CH}(\mathcal{F}_1, \ldots, \mathcal{F}_{d+1})$. Let $i \in [d+1]$ be a color whose class $\mathcal{F}_i$ has a non-empty intersection (by Theorem 6). Then one of the following statements must also hold:*

1. *an additional color class $\mathcal{F}_j$, for $j \in [d+1] \setminus \{i\}$ can be pierced by $f(d)$ points, or*
2. *the entire family $\mathcal{F}$ can be crossed by $g(d)$ lines.*

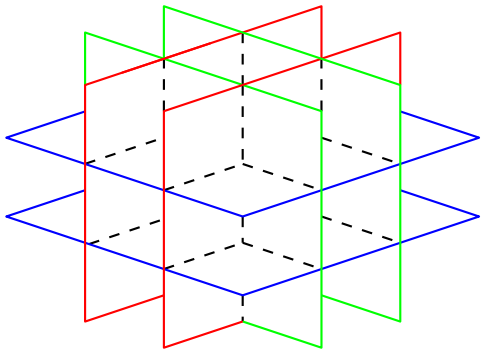


Figure 1 Optimality of the Colorful Helly Theorem in $\mathbb{R}^3$. For each $1 \leq i \leq 3$ the family $\mathcal{F}_i$ consists of x_i-orthogonal planes.

Theorem 8 is equivalent to the following statement concerning d-colored families of convex sets.

▶ **Theorem 9.** *For each dimension $d \geq 2$ there exist numbers $f'(d)$ and $g'(d)$ with the following property. Let $\mathcal{F}$ be a finite d-colored family of convex sets in $\mathbb{R}^d$, with color classes $\mathcal{F}_1, \ldots, \mathcal{F}_d$, that satisfies $\mathcal{CH}(\mathcal{F}_1, \ldots, \mathcal{F}_d)$. Then one of the following statements holds:*
1. *there is a color class $\mathcal{F}_j$, for $j \in [d]$, that can be pierced by $f'(d)$ points, or*
2. *the entire family $\mathcal{F}$ can be crossed by $g'(d)$ lines.*

Theorem 8 immediately follows from Theorem 9 by setting $f(d) = f'(d)$ and $g(d) = g'(d) + 1$. For the other direction, by letting $\mathcal{F}_{d+1} = \{\mathbb{R}^d\}$ we can set $f'(d) = f(d)$ and $g'(d) = g(d)$.

Notice that in the d-colored scenario of Theorem 9 one can use Theorem 6 to obtain *one* color class $\mathcal{F}_i$ that can be crossed by a single line (through a generic projection of $\mathcal{F}_d$ to $\mathbb{R}^{d-1}$). The main strength of Theorem 9 is that it shows a complementary relation between transversals to multiple colors $\mathcal{F}_i$, for $i \in [d]$. This relation can be further generalized as follows.

▶ **Theorem 10.** *For all $1 \leq i \leq d$ there exist numbers $f(i,d)$ and $g(i,d)$ with the following property. Let $\mathcal{F}$ be a finite $(d+1)$-colored family of convex sets in $\mathbb{R}^d$ that satisfies $\mathcal{CH}(\mathcal{F}_1, \ldots, \mathcal{F}_{d+1})$. Then there exist $k \in [d]$ and a re-labeling of the color classes $\mathcal{F}_1, \ldots, \mathcal{F}_{d+1}$ of $\mathcal{F}$ so that*
1. *$\bigcup_{1 \leq j \leq k} \mathcal{F}_j$ can be pierced by $f(k,d)$ points, and*
2. *$\bigcup_{k < j \leq d+1} \mathcal{F}_j$ can be crossed by $g(k,d)$ k-flats.*

In other words, Theorem 10 characterizes the families of sets with the Colorful Helly property up to their transversal structure by flats.

This paper is organized as follows. In Section 2 we prove our main technical results – Theorems 9 and 10. To this end, we establish a series of claims of independent interest that concern 2-*colored families* of convex sets. Despite the apparent weakness of the 2-colored hypothesis in dimension higher than 2, these results provide all the essential ingredients for our analysis. Theorem 9 is finally established by repeatedly invoking a so called "Step-Down" Lemma which provides a crucial relation between k-flat and $(k-1)$-flat transversals of families with the Colorful Helly property, for all $1 \leq k \leq d-1$.

The proof of the "Step-Down" Lemma is deferred to Section 3, and it is based on a careful adaptation of the machinery of Alon and Kleitman [4] and Alon and Kalai [2], to families of convex sets whose intersection graph is complete bi-partite.

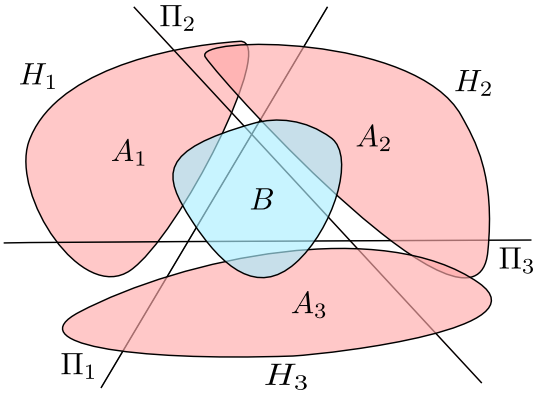


Figure 2 Proof of Lemma 11. We have $A_1, A_2, A_3 \in \mathcal{A}$ and $B \in \mathcal{B}$. Since $A_1 \cap A_2 \cap A_3 = \emptyset$, we have have halfspaces $H_i \supset A_i$, for $1 \leq i \leq 3$, so that $\bigcap_{i=1}^{n} H_i = \emptyset$. Hence, the set $B \in \mathcal{B}$ must cross at least one of the respective bounding lines Π_1 and Π_2 of H_1 and H_2 to meet the sets A_1, A_2 and A_3.

Section 4 is devoted to constructing a lower bound for g' in Theorem 9. Our example implies that, independently of the value given to $f'(d)$, $g'(d) \geq \lceil \frac{d+1}{2} \rceil$.

Finally, in Section 5 we conclude the paper with several intriguing questions for future study.

2 Proofs of Theorems 9 and 10

A crucial ingredient of our proof is the following claim which concerns 2-colored families.

▶ **Lemma 11.** *Let $\mathcal{A}$ and $\mathcal{B}$ be families of convex sets in $\mathbb{R}^d$ so that $A \cap B \neq \emptyset$ for every $A \in \mathcal{A}$ and $B \in \mathcal{B}$. Then either*

(1) $\bigcap \mathcal{A} \neq \emptyset$, *or*

(2) $\mathcal{B}$ *can be crossed by d hyperplanes.*

One can establish Theorem 9 in dimension $d = 2$ (with $f'(2) = 1$ and $g'(2) \leq 4$) by applying Lemma 11 twice. The weaker transversal guarantee of Lemma 11 in higher dimension $d \geq 3$ (namely, crossing by few hyperplanes instead of few lines) is due to the weaker, 2-colored hypothesis.

Proof of Lemma 11. Assume that *(1)* does not hold. Then by Helly's theorem there are convex sets $A_i \in \mathcal{A}$, for $1 \leq i \leq d+1$, with empty intersection. By a standard argument (see e.g. [8, Theorem 7.1]), there exist $d+1$ halfspaces $H_i \supseteq A_i$ with empty intersection. Let Π_i be the bounding hyperplane of H_i. We claim that the union of the first d hyperplanes Π_i (for $i = 1, \ldots, d$) must meet all the sets from B. See Figure 2 for an illustration in $\mathbb{R}^2$.

Indeed, consider the arrangement of $\Pi_1, \ldots, \Pi_d$ and suppose that a set $B \in \mathcal{B}$ does not intersect any of the hyperplanes Π_i. Then B must be completely contained in an open cell σ of their arrangement. Since B intersects each of the sets A_i, for $1 \leq i \leq d$, we obtain $\sigma = \bigcap_{i=1}^{d} H_i$. However, then B cannot intersect $A_{d+1} \subset H_{d+1}$, since $H_1 \cap \cdots \cap H_{d+1} = \emptyset$. This contradiction implies *(2)*. ◀

Both Theorems 9 and 10 are established by iterating the following more refined variant of Lemma 11.

▶ **Lemma 12** ("Step-Down" Lemma). *For any $1 \leq k \leq d$ and $m \geq 1$ there exist numbers $F(m,k,d)$ and $G(m,k,d)$ with the following property.*

Let $\mathcal{A}$ and $\mathcal{B}$ be finite families of convex sets in $\mathbb{R}^d$ so that the family

$$\mathcal{I}(\mathcal{A},\mathcal{B}) := \{A \cap B \mid A \in \mathcal{A}, B \in \mathcal{B}\}$$

can be crossed by m k-flats. Then one of the following conditions is satisfied:

1. *$\mathcal{A}$ can be pierced by $F(m,k,d)$ points, or*
2. *$\mathcal{B}$ can be crossed by $G(m,k,d)$ $(k-1)$-flats.*

Notice that the hypothesis of Lemma 12 implies, in particular, that every two sets $A \in \mathcal{A}, B \in \mathcal{B}$ intersect. Thus, Lemma 11 deals with the special case of Lemma 12 in which $k = d$, yielding $F(1,d,d) = 1$ and $G(1,d,d) \leq d$.

We defer the somewhat complex proof of Lemma 12 to Section 3. It combines the standard duality relation between transversal and packing numbers of hypergraphs with a "hyperplane" variant of Theorem 4, due to Alon and Kalai [2], in which we are given a collection of hyperplanes H and seek to find a small hyperplane transveral to all the convex sets that are crossed by a fixed fraction of the hyperplanes of H.

We are now ready to establish Theorem 9.

Proof of Theorem 9. Let $\mathcal{F}$ be a d-colored family that satisfies $\mathcal{CH}(\mathcal{F}_1,\ldots,\mathcal{F}_d)$ and does not satisfy conclusion 1. Since the labeling of the color classes $\mathcal{F}_1,\ldots,\mathcal{F}_d$ is arbitrary, it suffices to show that the last family $\mathcal{F}_d$ can be crossed by few lines.

The underlying idea of our analysis is as follows. We apply the "Step-Down" Lemma 12 $d-1$ times. In the i-th iteration (for $1 \leq i \leq d-1$) we deal with a $(d-i+1)$-colored and *essentially* $(d-i+1)$-*dimensional* scenario in which the family of all the $(d-i+1)$-wise intersections

$$\mathcal{I}(\mathcal{F}_i,\ldots,\mathcal{F}_d) := \left\{ \bigcap_{j=i}^{d} A_j \mid A_j \in \mathcal{F}_j \right\}$$

is "captured" by only $M = M(i,d)$ copies of $\mathbb{R}^{d-i+1}$ within $\mathbb{R}^d$. Unless $\mathcal{F}_i$ can be pierced by $F(M,i,d)$ points, the "Step-Down" Lemma can be used to further reduce the intrinsic "transversal dimension" of the remaining sets $\mathcal{F}_{i+1},\ldots,\mathcal{F}_d$ to $d-i$.

For reasons that will become evident shortly, we set

$$M(i,d) := \begin{cases} 1 & \text{for } i = 1, \\ d & \text{for } i = 2, \\ G(M(i-1,d), d-i+2, d) & \text{for } 3 \leq i \leq d-1. \end{cases}$$

For $i = 1$, the condition that $\mathcal{I}(\mathcal{F}_1,\ldots,\mathcal{F}_d)$ is crossed by $\mathbb{R}^d$ is equivalent to the $\mathcal{CH}(\mathcal{F}_1,\ldots,\mathcal{F}_d)$ hypothesis. Notice that the families $\mathcal{A} := \mathcal{F}_1$ and $\mathcal{B} := \mathcal{I}(\mathcal{F}_2,\ldots,\mathcal{F}_d)$ satisfy the hypothesis of Lemma 11. Therefore, unless $\mathcal{F}_1$ can be pierced by a single point, the family $\mathcal{I}(\mathcal{F}_2,\ldots,\mathcal{F}_d)$ can be crossed by $M(2,d) = d$ hyperplanes.

Let us now fix $2 \leq i \leq d-1$ and assume that $\mathcal{I}(\mathcal{F}_i,\ldots,\mathcal{F}_d)$ can be crossed by $M = M(i,d)$ $(d-i+1)$-flats. Note that the families $\mathcal{A} := \mathcal{F}_i$ and $\mathcal{B} := \mathcal{I}(\mathcal{F}_{i+1},\ldots,\mathcal{F}_d)$ satisfy the 2-colored hypothesis of Lemma 12. Therefore, given that $\mathcal{F}_i$ cannot be pierced by $F(M,d-i+1,d)$ points, the other family $\mathcal{I}(\mathcal{F}_{i+1},\ldots,\mathcal{F}_d)$ can be crossed by $M(i+1,d) = G(M,d-i+1,d)$ $(d-i)$-flats.

Assuming neither of the families $\mathcal{F}_i$, for $1 \leq i \leq d-1$, can be pierced by $F(M(i,d), d-i+1, d)$ points, by the end of the $(d-1)$-st iteration we can cross the last color class $\mathcal{F}_d$ by $G(M(d-1,d),2,d)$ lines.

This proves Theorem 9 with

$$f'(d) = \max\{F(M(i,d), d-i+1, d) \mid 1 \leq i \leq d-1\}, \text{ and}$$
$$g'(d) = d \cdot G(M(d-1,d),2,d). \quad \blacktriangleleft$$

▶ Remark. In the proof of Theorem 9, the value of $g'(d)$ can be further improved to

$$g'(d) = (d-1) \cdot G(M(d-1,d),2,d) + 1$$

by observing that at least one of the families $\mathcal{F}_1, \ldots, \mathcal{F}_d$ can be crossed by a *single* line. To this end, we project $\mathcal{F}$ in a generic direction $\vec{\nu}$ and apply Theorem 6 to the resulting d-colored family $\mathcal{F}(\vec{\nu}) = \mathcal{F}_1(\vec{\nu}) \cup \cdots \cup \mathcal{F}_d(\vec{\nu})$ within $\mathbb{R}^{d-1}$. This yields an intersecting color class $\mathcal{F}_i(\vec{\nu})$ within $\mathbb{R}^{d-1}$ and, therefore, a $\vec{\nu}$-parallel line which crosses the respective color class $\mathcal{F}_i$.

Proof of Theorem 10. The Theorem is obviously true for $d = 1$ (with $f(1,1) = 1$, $g(1,1) = 1$). Assume with no loss of generality that the last color class $\mathcal{F}_{d+1}$ can be pierced by a point (in accordance with Theorem 6). We adopt the notation of the previous proof while dealing with the remaining color classes $\mathcal{F}_1, \ldots, \mathcal{F}_d$.

Let l be the size of the largest sequence $j_1, j_2 \ldots, j_l$ so that no class $\mathcal{F}_{j_i}$ can be pierced by $F(M(l,d), d-l+1, d)$ points. Let $\mathcal{F}'$ be the relabeling of $\mathcal{F}$ whose first l color classes satisfy $\mathcal{F}'_i = \mathcal{F}_{j_i}$, for $1 \leq i \leq l$. By following the first $l-1$ iterations of the proof of Theorem 9, we obtain that $\mathcal{F}'_l = \mathcal{F}_{j_l}$ can be crossed by $G(M(l-1,d), d-l, d)$ $(d-l+1)$-flats. By reordering of $j_1, \ldots, j_l$, this establishes the claim of Theorem 10 for $\mathcal{F}$ with

$$k = d-l+1,$$
$$f(k,d) = k \cdot F(M(d-k+1,d), k, d), \text{ and}$$
$$g(k,d) = (d-k+1) \cdot G(M(d-k,d), k+1, d). \quad \blacktriangleleft$$

3 Proof of the "Step-Down" Lemma

We develop a bi-partite variant of the machinery that was used by Alon and Kleitman [4] to establish the (p,q)-Conjecture (Theorem 3). This method was extended by Alon and Kalai [2] to obtain an analogous result for hyperplane transversals.

3.1 From piercing to packing numbers

The crucial ingredient of Alon-Kleitman approach was a duality relation between transversal (or piercing), and packing (or matching) numbers of hypergraphs.

▶ **Definition 13.** Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a hypergraph, where $\mathcal{V}$ is a finite set of elements and $\mathcal{E}$ is a family of subsets of $\mathcal{V}$. The elements of $\mathcal{V}$ are called *vertices*, and the sets of $\mathcal{E}$ are called *edges*.

A subset $A \subset \mathcal{V}$ is a *transversal* for $\mathcal{G}$ if it intersects every edge $S \in \mathcal{E}$ (i.e., $A \cap S \neq \emptyset$ for each $S \in \mathcal{E}$). The *transversal number* $\tau(\mathcal{G})$ of $\mathcal{G}$ is the size $|A|$ of the smallest such transversal A.

A non-negative function $f : \mathcal{V} \to \mathbb{R}$ is a *fractional transversal* for $\mathcal{G}$ if it satisfies $\sum_{x \in S} f(x) \geq 1$ for every edge $S \in \mathcal{E}$. The *fractional transversal number* $\tau^*(\mathcal{G})$ of $\mathcal{G}$ is the

total "weight" $\sum_{x\in\mathcal{V}} f(x)$ of the "lightest" fractional transversal f of $\mathcal{G}$ (that is, it is the smallest possible value $\sum_{x\in\mathcal{V}} f(x)$ that can be attained by a fractional tranfsversal f).

A subset of edges $\mathcal{E}' \subseteq \mathcal{E}$ is called a *b-matching* (or *b-packing*) for $\mathcal{G}$ if every vertex $x \in \mathcal{V}$ belongs to at most b edges of $\mathcal{E}'$. The *b-matching number* $\nu_b(\mathcal{G})$ of $\mathcal{G}$ is the size $|\mathcal{E}'|$ of the largest such b-matching $\mathcal{E}'$.

A non-negative function $g : \mathcal{E} \to \mathbb{R}$ is a *fractional matching* for $\mathcal{G}$ if it satisfies

$$\sum_{S\in\mathcal{E}:x\in S} g(S) \leq 1$$

for every $x \in \mathcal{V}$. The *fractional matching number* $\nu^*(\mathcal{G})$ of $\mathcal{G}$ is the total "weight" $\sum_{S\in\mathcal{E}} g(S)$ of the "heaviest" fractional matching g of $\mathcal{G}$ (that is, it is the largest possible value $\sum_{S\in\mathcal{E}} g(S)$ that can be attained by a fractional matching g).

A standard use of Linear Programming duality [4, 2, 3] yields the following relation between transversal and matching numbers of $\mathcal{G}$.

▶ **Theorem 14.** *We have*

$$\nu_b(\mathcal{G})/b \leq \nu^*(\mathcal{G}) = \tau^*(\mathcal{G}) \leq \tau(\mathcal{G})$$

for every hypergraph $\mathcal{G}$ and $b \geq 1$.

The proof of Theorem 3 by Alon and Kleitman [4] combines the following key elements:

- An abstract hypergraph $\mathcal{G}_0(\mathcal{F})$, whose edges correspond to the sets of $\mathcal{F}$, is constructed. Each vertex of $\mathcal{G}_0(\mathcal{F})$ is a point that pierces some sub-family $\mathcal{F}' \subset \mathcal{F}$. (To keep the vertex set finite, we add *one* vertex for each $\mathcal{F}' \subset \mathcal{F}$ with non-empty intersection $\bigcap \mathcal{F}' \neq \emptyset$.)
- The fractional matching number $\nu^*(\mathcal{G}_0(\mathcal{F})) = \tau^*(\mathcal{G}_0(\mathcal{F}))$ is bounded from above using a suitable fractional Helly-type result (Theorem 1).
- The fractional transversal for $\mathcal{G}_0(\mathcal{F})$ is converted to an integral one using a weak ϵ-net result for point transversals [1].

3.1.1 Overview

As we cast the 2-colored setup of the "Step-Down" Lemma into the above abstract framework, several fundamental challenges are to be addressed.

As we seek a *relation* between the transversal numbers of $\mathcal{A}$ and $\mathcal{B}$, we maintain *two hypergraphs* $\mathcal{G}_0(\mathcal{A})$ and $\mathcal{G}_{k-1}(\mathcal{B})$, where the former (resp., latter) hypergraph describes partial point (resp., $(k-1)$-flat) transversals to $\mathcal{A}$ (resp., $\mathcal{B}$). To show that at least one of $\mathcal{G}_0(\mathcal{A})$ and $\mathcal{G}_{k-1}(\mathcal{B})$ has a bounded fractional packing number, we need a suitable fractional Helly-type result which is conveniently provided by the fractional variant of our 2-colored Lemma 11. Finally, to convert a fractional transversal for $\mathcal{G}_{k-1}(\mathcal{B})$ into an integral one, we need a small-size weak ϵ-net construction for $(k-1)$-flats.

Unfortunately, no Helly-type results and no weak ϵ-net constructions are known for transversals by general $(k-1)$-flats in $\mathbb{R}^d$, unless $k = 1$ [4] or $k = d$ [2]. Note though that, in the scenario of Lemma 12, the pairwise intersections $\mathcal{I}(\mathcal{A}, \mathcal{B})$ are assumed to "occur" within few k-dimensional flats of $\mathbb{R}^d$. We can therefore invoke the fractional variant of Lemma 11 in dimension k and similarly apply the weak ϵ-net construction of Alon and Kalai [2] for hyperplanes in $\mathbb{R}^k$.

3.2 Bounding the fractional packing number

Let $\mathcal{A}$ and $\mathcal{B}$ be families of convex sets that satisfy the hypothesis of Lemma 12. That is, the family $\mathcal{I}(\mathcal{A}, \mathcal{B})$ of pairwise intersections can be crossed by m k-flats $\Gamma_1, \ldots, \Gamma_m$.

3.2.1 The hypergraphs $\mathcal{G}_0(\mathcal{A})$ and $\mathcal{G}_{k-1}(\mathcal{B})$

Below we define the abstract hypergraphs $\mathcal{G}_0(\mathcal{A})$ and $\mathcal{G}_{k-1}(\mathcal{B})$ which describe, respectively, partial point transversals to $\mathcal{A}$, and partial transversals by $(k-1)$-flats to $\mathcal{B}$.

The hypergraph $\mathcal{G}_0(\mathcal{A}) = (\mathcal{V}_\mathcal{A}, \mathcal{E}_\mathcal{A})$ is constructed analogously to the one of Alon and Kleitman [4]: For every subfamily $\mathcal{A}' \subset \mathcal{A}$ with $\bigcap \mathcal{A}' \neq \emptyset$ we add a point $x_{\mathcal{A}'} \in \bigcap \mathcal{A}'$ to $\mathcal{V}_\mathcal{A}$, and for every convex set $A \in \mathcal{F}$ we add the edge $e_A := \{x_{\mathcal{A}'} \mid \bigcap \mathcal{A}' \neq \emptyset, A \in \mathcal{A}'\}$ to $\mathcal{E}_\mathcal{A}$.

The definition of $\mathcal{G}_{k-1}(\mathcal{B}) = (\mathcal{V}_\mathcal{B}, \mathcal{E}_\mathcal{B})$ is somewhat more involved: For every subfamily $\mathcal{B}' \subset \mathcal{B}$ that can be crossed by a $(k-1)$-flat *within* $\bigcup_{i=1}^m \Gamma_i$, we add one such $(k-1)$-flat $\sigma_{\mathcal{B}'} \subset \bigcup_{i=1}^m \Gamma_i$ to $\mathcal{V}_\mathcal{B}$. Accordingly, each $B \in \mathcal{B}$ yields the edge

$$e_B := \{\sigma_{\mathcal{B}'} \mid B \in \mathcal{B}'\} \in \mathcal{E}_\mathcal{B}.$$

To show that *at least one* of the hypergraphs $\mathcal{G}_0(\mathcal{A})$ or $\mathcal{G}_{k-1}(\mathcal{B})$ has a bounded fractional packing number, we use the following fractional variant of our 2-colored Lemma 11.

▶ **Lemma 15** (Fractional 2-colored Lemma). *For every $0 < \alpha \leq 1$ and $d \geq 1$ there exist $\gamma = \gamma(\alpha, d)$ and $\lambda = \lambda(\alpha, d)$ with the following property. Let $\mathcal{A}$ and $\mathcal{B}$ be finite (multi-)families of convex sets in $\mathbb{R}^d$ so that $A \cap B \neq \emptyset$ holds for at least $\alpha|\mathcal{A}||\mathcal{B}|$ of the pairs $A \in \mathcal{A}$ and $B \in \mathcal{B}$. Then either*

1. *one can pierce at least $\gamma|\mathcal{A}|$ members of $\mathcal{A}$ by a single point, or*
2. *one can cross at least $\lambda|\mathcal{B}|$ members of $\mathcal{B}$ by a single hyperplane.*

For a proof of Lemma 15, we refer the reader to Section 3.2.1 of the full version of the paper [17]. Here we prove the following auxiliary statement.

▶ **Claim 16.** *We have either $\nu^*(\mathcal{G}_0(\mathcal{A})) \leq 1/(\gamma(1/m, k))$ or $\nu^*(\mathcal{G}_{k-1}(\mathcal{B})) \leq 1/(\lambda(1/m, k))$, where m, $\mathcal{G}_0(\mathcal{A})$ and $\mathcal{G}_{k-1}(\mathcal{B})$ are as defined above, and the functions γ and λ are defined as in Lemma 15.*

Proof of Claim 16. The fractional packing and fractional transversal numbers exist as we are optimizing continuous functions on a compact set. Moreover, the optimal value may be obtained via a rational approximation. Thus, given the contrapositive assumption, we have a pair of non-negative rational assignments $f : \mathcal{E}_\mathcal{A} \to \mathbb{Q}$ and $g : \mathcal{E}_\mathcal{B} \to \mathbb{Q}$ so that the following inequalities hold for all $x_0 \in \mathcal{V}_\mathcal{A}$ and $\sigma_0 \in \mathcal{V}_\mathcal{B}$:

$$\sum_{x_0 \in e} f(e) < \gamma(1/m, k) \sum_{e \in \mathcal{E}_\mathcal{A}} f(e), \tag{1}$$

$$\sum_{\sigma_0 \in e} g(e) < \lambda(1/m, k) \sum_{e \in \mathcal{E}_\mathcal{B}} g(e). \tag{2}$$

By scaling f and g, we end up with a pair of integer functions $f : \mathcal{E}_\mathcal{A} \to \mathbb{Z}^+$ and $g : \mathcal{E}_\mathcal{B} \to \mathbb{Z}^+$ which still satisfy the Inequalities (1) and (2). By the definition of $\mathcal{G}_0(\mathcal{A})$ and $\mathcal{G}_{k-1}(\mathcal{B})$, this yields a pair of multisets $\hat{\mathcal{A}}$ and $\hat{\mathcal{B}}$ of, respectively, $\mathcal{A}$ and $\mathcal{B}$, so that

- **(i)** no point in $\mathbb{R}^d$ crosses more than $\gamma(1/m, k)|\hat{\mathcal{A}}|$ members of $\hat{\mathcal{A}}$, and
- **(ii)** no $(k-1)$-flat within $\bigcup_{i=1}^m \Gamma_i$ crosses more than $\lambda(1/m, k)|\hat{\mathcal{B}}|$ members of $\hat{\mathcal{B}}$.

By the pigeonhole principle, one of the k-flats Γ_i must cross at least $(1/m)|\mathcal{I}(\mathcal{A}, \mathcal{B})|$ of the pairwise intersections $\mathcal{I}(\mathcal{A}, \mathcal{B})$. Applying Lemma 15 to the cross-sections $\{A \cap \Gamma_i \mid A \in \hat{\mathcal{A}}\}$ and $\{B \cap \Gamma_i \mid B \in \hat{\mathcal{B}}\}$ within $\Gamma_i \cong \mathbb{R}^k$, and with $\alpha := 1/m$, yields the eventual contradiction to the above properties (i) and (ii) of $\hat{\mathcal{A}}$ and $\hat{\mathcal{B}}$. ◀

3.3 Wrap-up

Combining Claim 16 with Theorem 14, we obtain that at least one of the graphs $\mathcal{G}_0(\mathcal{A})$ and $\mathcal{G}_{k-1}(\mathcal{B})$ has a bounded *fractional* transversal number, so one of the following inequalities must hold:

$$\tau^*(\mathcal{G}_0(\mathcal{A})) \leq \frac{1}{\gamma(1/m,k)}, \qquad \tau^*(\mathcal{G}_{k-1}(\mathcal{B})) \leq \frac{1}{\lambda(1/m,k)}.$$

Analogously to the proof of Claim 16, we obtain respectively either a rational (and not everywhere zero) function $f : \mathcal{V}_\mathcal{A} \to \mathbb{Q}^+$ so that every edge $e \in \mathcal{E}_\mathcal{A}$ (representing some set $A \in \mathcal{A}$) contains vertices (i.e., points) of total weight

$$\sum_{x \in e} f(x) \geq \gamma(1/m,k) \sum_{x \in \mathcal{V}_\mathcal{A}} f(x),$$

or a similar function $g : \mathcal{V}_\mathcal{B} \to \mathbb{Q}^+$ so that every edge $e \in \mathcal{E}_\mathcal{B}$ contains vertices of total weight

$$\sum_{\sigma \in e} g(\sigma) \geq \lambda(1/m,k) \sum_{\sigma \in \mathcal{V}_\mathcal{B}} g(\sigma).$$

Arguing as in the proof of Claim 16, we obtain either (i) a multiset of points $\hat{\mathcal{V}}_\mathcal{A} \subset \mathbb{R}^d$ so that any member A of $\mathcal{A}$ contains at least $\gamma(1/m,k)|\hat{\mathcal{V}}_\mathcal{A}|$ of these points, or (ii) a multiset $\hat{\mathcal{V}}_\mathcal{B}$ of $(k-1)$-flats within $\bigcup_{i=1}^m \Gamma_i$ so that any member B of $\mathcal{B}$ is crossed by at least $\lambda(1/m,k)|\hat{\mathcal{V}}_\mathcal{B}|$ of the flats.

In the former case, we use Theorem 4 to show that, in case (i), the family $\mathcal{A}$ can be pierced by

$$F(m,k,d) := W\left(\gamma(1/m,k), 0, d\right)$$

points.

In the remaining case (ii), we use the following analogue of Theorem 4 for hyperplane transversals, due to Alon and Kalai [2]:

▶ **Lemma 17** (Weak ϵ-net for hyperplanes). *For any dimension $d \geq 1$ and $\epsilon > 0$ there is $W_{hpl}(\epsilon, d)$ with the following property: For every finite (multi-)set H of hyperplanes in $\mathbb{R}^d$ one can find $W_{hpl}(\epsilon, d)$ hyperplanes in $\mathbb{R}^d$ whose union crosses every convex set $A \subseteq \mathbb{R}^d$ that meets at least $\epsilon|H|$ of the hyperplanes of H.*

For each $1 \leq i \leq m$ we apply Lemma 17 to construct a weak $(\lambda(1/m,k)/m)$-net with respect to the $(k-1)$-flats $\sigma \in \hat{\mathcal{V}}_\mathcal{B}$ that are contained in $\Gamma_i \cong \mathbb{R}^k$. It is immediate to check that the resulting family of at most

$$G(m,k,d) := m \cdot W_{hpl}\left(\frac{\lambda(1/m,k)}{m}, k-1, k\right)$$

$(k-1)$-flats crosses each $B \in \mathcal{B}$: Since B is crossed by at least $\lambda(1/m,k)|\hat{\mathcal{V}}_\mathcal{B}|$ $(k-1)$-flats of $\hat{\mathcal{V}}_\mathcal{B}$, and at least $(1/m)\lambda(1/m,k)|\hat{\mathcal{V}}_\mathcal{B}|$ of such flats must be contained in some k-flat Γ_i, then B must be crossed by the corresponding k-dimensional net. □

4 A lower bound for Theorem 9

▶ **Theorem 18.** *For every $d \geq 2$ and integer $f \geq 1$ there exists a d-colored family $\mathcal{F} = \mathcal{F}_1 \uplus \mathcal{F}_2 \uplus \ldots \uplus \mathcal{F}_d$ in $\mathbb{R}^d$ that satisfies $\mathcal{CH}(\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_d)$ and the following additional properties:*

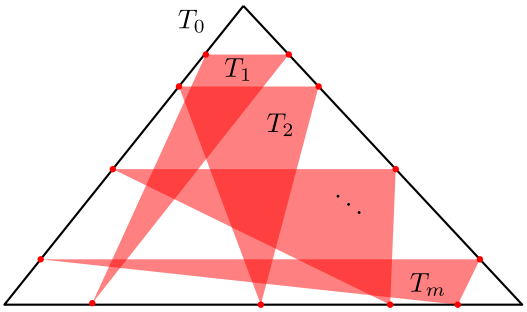


Figure 3 The planar construction (for $m = 4$). Each triangle T_i has a horizontal topmost side which lies below all the pairwise intersections of $T_1, \dots T_{i-1}$.

- *For every $1 \leq i \leq d$, one needs at least f points to pierce the color class $\mathcal{F}_i$. (In other words, $\tau(\mathcal{G}(\mathcal{F}_i)) \geq f$.)*
- *At least $\lceil \frac{d+1}{2} \rceil$ lines are necessary to cross $\bigcup \mathcal{F}_i$.*

We prove the result in the following two subsections. We begin with the case $d = 2$ which is later used to deal with the general case.

4.1 The planar construction

Let $m = 2f$ and T_0 be a triangle in the plane so that its bottom side is parallel to the x-axis. We first construct m triangles $T_1, \dots, T_m$, each with one horizontal side and vertices in the relative interiors of the three sides of T_0, and such that no three of these triangles T_i, T_j, T_k for $1 \leq i < j < k \leq m$ have a common intersection. A way to do this is to construct them recursively: we start with two arbitrary such triangles T_1 and T_2 and at each step $i > 2$ we place the horizontal side of T_i sufficiently close to the horizonal side of T_0 so that it avoids all previous pairwise intersections (see Figure 3). Let the first color class $\mathcal{F}_1$ be the resulting family $\{T_1, \dots, T_m\}$. Clearly we need at least $m/2 = f$ points to pierce $\mathcal{F}_1$.

Let E_1, E_2, E_3 be the three sides of T_0. As each set of $\mathcal{F}_1$ intersects the relative interior of each E_i, for $1 \leq i \leq 3$, we can slightly shrink each E_i away from its adjacent vertices of T_0 while preserving the intersection with every element of $\mathcal{F}_1$. The family $\mathcal{F}_2$ will consist of m slightly translated copies of each (previously shrunk) segment E_i so that they still intersect every triangle in $\mathcal{F}_1$ but are still pairwise disjoint. Note that we need at least $3m > f$ points to pierce $\mathcal{F}_2$.

In order to cross $\mathcal{F}_1 \cup \mathcal{F}_2$ with lines, we need in particular to cross the interiors of E_1, E_2, E_3, so at least 2 lines are needed.

4.2 The general construction

Set $d > 2$ and $m = 2f$. Let $\Delta^{(d)} \subset \mathbb{R}^d$ be a d-simplex with vertex set $V = \{v_1, v_2, \dots, v_{d+1}\}$. For each $1 \leq i \leq d-1$, define τ_i to be the triangle with vertices $\{v_i, v_{i+1}, v_{i+2}\}$. As in the planar case, let $\mathcal{T}_i$ be a family of m triangles, each with vertices in the relative interiors of the three sides of τ_i, such that no three of them intersect. Let $\hat{\mathcal{F}}_i^{(d)}$ be the family consisting of the sets

$$\operatorname{conv}((V \setminus \{v_i, v_{i+1}, v_{i+2}\}) \cup \tau),$$

with $\tau \in \mathcal{T}_i$. Let $\hat{\mathcal{F}}_d^{(d)}$ denote the family of all the $(d-1)$-dimensional faces (facets) of $\Delta^{(d)}$; see Figure 4.

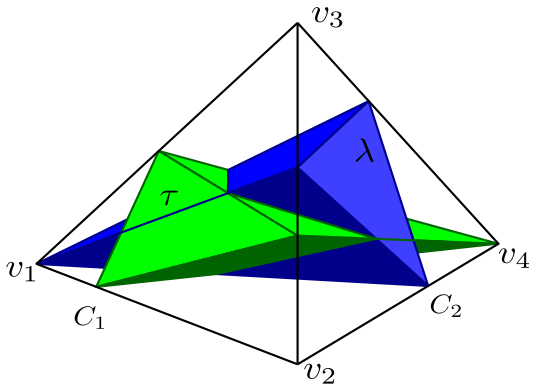


Figure 4 The construction of $\hat{\mathcal{F}}^{(3)}$ – a pair of sets $C_1 \in \hat{\mathcal{F}}_1^{(3)}$ and $C_2 \in \hat{\mathcal{F}}_2^{(3)}$ are depicted. We have $C_1 = \mathrm{conv}(\tau, v_4)$ and $C_2 = \mathrm{conv}(\lambda, v_1)$, with $\tau \in \mathcal{T}_1$ and $\lambda \in \mathcal{T}_2$. The sets of $\hat{\mathcal{F}}_3^{(3)}$ are the facets of the bounding simplex $\Delta^{(3)}$.

As the resulting d-colored family $\hat{\mathcal{F}}^{(d)} = \bigcup \hat{\mathcal{F}}_i^{(d)}$ can obviously be pierced by $d+1$ points, the convex sets in $\hat{\mathcal{F}}$ have to be suitably shrunk in order to satisfy the conditions of Theorem 18. However, before we describe the actual family $\mathcal{F}^{(d)}$, we establish a key property of the families $\hat{\mathcal{F}}_i^{(d)}$. We provide the proof of the following lemma in Section 4.2 of the full version of the paper [17].

▶ **Lemma 19.** *For any selection of $C_i \in \hat{\mathcal{F}}_i^{(d)}$ with $2 \le i \le d$ we have*

$$\left(\bigcap_{i=1}^{d-1} C_i\right) \cap \mathrm{relint}(C_d) \neq \emptyset,$$

where $\mathrm{relint}(C)$ *denotes the relative interior of* C.

We are almost done with the construction. In view of Lemma 19, we may shrink all the elements of $\hat{\mathcal{F}}^{(d)}$ away from the $(d-2)$-dimensional faces of $\Delta^{(d)}$ in such a way that they remain convex and the colorful intersections continue to be non-empty. In this way we obtain the families $\mathcal{F}_1^{(d)}, \ldots, \mathcal{F}_{d-1}^{(d)}$. To construct the last family $\mathcal{F}_d^{(d)}$, we take an additional step: we take m parallel copies of each so that they still intersect every element of $\mathcal{F}_1 \cup \cdots \cup \mathcal{F}_{d-1}$ but are pairwise disjoint.

By the cut-off procedure, no three sets of the same $\mathcal{F}_i$ intersect for $i \in [d-1]$ (as any such intersection would project to a triple intersection within $\mathcal{T}_i$). Thus, in order to pierce any such $\mathcal{F}_i$ at least $\frac{m}{2} = f$ points are needed. To cross $\mathcal{F} = \mathcal{F}_1 \cup \cdots \cup \mathcal{F}_d$ by lines we also need to cross the relative interiors of the facets of Δ. No line can pierce more than two such interiors. Therefore, at least $\lceil \frac{d+1}{2} \rceil$ lines are needed. This concludes the proof of Theorem 18.

5 Discussion

We studied families of convex sets which satisfy the Colorful Helly hypothesis. Our Theorems 8 and 10 offer complementary relations between the "transversal dimensions" of individual color classes.

We conjecture that an even stronger phenomenon happens:

▶ **Conjecture 20.** *For all $1 \leq k \leq d$ there exist numbers $h(k,d)$ with the following property. For any d-colored family $\mathcal{F}$ of convex sets in $\mathbb{R}^d$ with $\mathcal{CH}(\mathcal{F}_1, \ldots, \mathcal{F}_d)$ there exist numbers $k_1, \ldots, k_d$ so that*

1. *$\sum_{1 \leq i \leq d} k_i \leq d$, and*
2. *each color class $\mathcal{F}_i$, for $i \in [d]$, can be crossed by $h(k_i, d)$ k_i-flats.*

It is easy to check that Conjecture 20 is sharp for families of flats. The most elementary instance of the conjecture arises for $d = 3$ and $\mathcal{F}_3 = \{\mathbb{R}^3\}$. The remaining two classes $\mathcal{F}_1$ and $\mathcal{F}_2$ satisfy a 2-colored hypothesis. If one of the classes has a transversal by few points, then Conjecture 20 holds for the families, as the other class can simply be pierced by $\mathbb{R}^3$. Otherwise, by Lemma 11 both $\mathcal{F}_1$ and $\mathcal{F}_2$ can be pierced by few planes. Then the validity of Conjecture 20 in this case depends on the answer to the following question:

▶ **Problem.** *Is it true that for any two families $\mathcal{A}, \mathcal{B}$ of convex sets in $\mathbb{R}^3$ so that $A \cap B \neq \emptyset$ holds for all $A \in \mathcal{A}$ and $B \in \mathcal{B}$, one of the families $\mathcal{A}$ or $\mathcal{B}$ can be crossed by $O(1)$ lines?*

Another intriguing question is what are the "true" values of $f'(d)$ and $g'(d)$ for Theorem 9 or, more precisely, what is the relation between these parameters? For example, does the theorem still hold with $f'(d) = 1$ and large enough $g'(d)$, as it happens for $d = 2$?

References

1. N. Alon, I. Bárány, Z. Füredi, and D. J. Kleitman. Point selections and weak ε-nets for convex hulls. *Combinatorics Probability and Computing*, 1(3):189–200, 1992.
2. N. Alon and G. Kalai. Bounding the piercing number. *Discrete & Computational Geometry*, 13(3):245–256, 1995.
3. N. Alon, G. Kalai, J. Matoušek, and R. Meshulam. Transversal numbers for hypergraphs arising in geometry. *Advances in Applied Mathematics*, 29(1):79–101, 2002.
4. N. Alon and D. J. Kleitman. Piercing convex sets and the Hadwiger-Debrunner (p, q)-problem. *Advances in Mathematics*, 96(1):103–112, 1992.
5. N. Amenta, J. A. De Loera, and P. Soberón. Helly's Theorem: New Variations and Applications. *Contemporary Mathematics*, 685:55–95, 2017. URL: `https://arxiv.org/abs/1508.07606`.
6. J. L. Arocha, I. Bárány, J. Bracho, R. Fabila, and L. Montejano. Very colorful theorems. *Discrete & Computational Geometry*, 42(2):142–154, 2009.
7. I. Bárány. A generalization of Carathéodory's theorem. *Discrete Mathematics*, 40(2-3):141–152, 1982.
8. I. Bárány. Tensors, colours, octahedra. In *Geometry, Structure and Randomness in Combinatorics*, pages 1–17. Springer, 2014.
9. L. Danzer. Über ein Problem aus der kombinatorischen Geometrie. *Archiv der Mathematik*, 8(5):347–351, 1957.
10. L. Danzer, B. Grünbaum, and V. Klee. *Helly's theorem and its relatives*. Proceedings of symposia in pure mathematics: Convexity. American Mathematical Society, 1963.
11. J. Eckhoff. *Handbook of Convex Geometry*, chapter Helly, Radon, and Carathéodory type theorems, pages 389–448. Elsevier, 1990.
12. J. E. Goodman, R. Pollack, and R. Wenger. *New Trends in Discrete and Computational Geometry*, chapter Geometric Transversal Theory, pages 163–198. Springer Berlin Heidelberg, Berlin, Heidelberg, 1993.
13. H. Hadwiger and H. Debrunner. Über eine Variante zum Helly'schen Satz. *Arch. Math.*, 8:309–313, 1957.

14 E. Helly. Über Mengen konvexer Körper mit gemeinschaftlichen Punkte. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 32:175–176, 1923.

15 A. F. Holmsen, J. Pach, and H. Tverberg. Points surrounding the origin. *Combinatorica*, 28(6):633–644, 2008.

16 M. Katchalski and A. Liu. A problem of geometry in R^n. *Proceedings of the American Mathematical Society*, 75(2):284–288, 1979.

17 Leonardo Martínez-Sandoval, Edgardo Roldán-Pensado, and Natan Rubin. Further consequences of the colorful helly hypothesis. *arXiv preprint*, 2018. URL: `https://arxiv.org/abs/1803.06229`.

18 L. Santaló. Un teorema sobre conjuntos de paralelepípedos de aristas paralelas. *Publicaciones del Instituto de Matemáticas, Universidad Nacional del Litoral*, II(4):49–60, 1940.

19 R. Wenger and A. Holmsen. *The Handbook of Discrete and Computational Geometry*, chapter Helly-type Theorems and Geometric Transversals, pages 91–124. Chapman and Hall/CRC, third edition edition, 2017.

Random Walks on Polytopes of Constant Corank

Malte Milatz
ETH Zürich
Switzerland
mmilatz@inf.ethz.ch

Abstract

We show that the pivoting process associated with one line and n points in r-dimensional space may need $\Omega(\log^r n)$ steps in expectation as $n \to \infty$. The only cases for which the bound was known previously were for $r \leq 3$. Our lower bound is also valid for the expected number of pivoting steps in the following applications: (1) The Random-Edge simplex algorithm on linear programs with n constraints in $d = n - r$ variables; and (2) the directed random walk on a grid polytope of corank r with n facets.

2012 ACM Subject Classification Theory of computation → Randomness, geometry and discrete structures, Theory of computation → Linear programming

Keywords and phrases polytope, unique sink orientation, grid, random walk

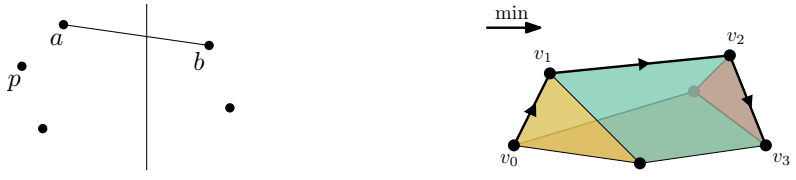
Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.60

Related Version The full version with appendix may be obtained from `https://arxiv.org/abs/1802.09324`.

Acknowledgements I would like to thank Bernd Gärtner, Ahad N. Zehmakan, Jerri Nummenpalo and Alexander Pilz for many useful suggestions and discussions.

1 Introduction

In 2001 Gärtner et al. [6] introduced a random process involving a vertical line and n points in the plane (see Figure 1a) which they called "the fast process" or just "one line and n points". The starting position of this process is a pair $\{a, b\}$ of points that lie on opposite sides of the vertical line. In each step the process picks a point p (a pivot) uniformly at random from the points that lie below the non-vertical line ab. The subsequent position is then the unique pair $\{p, q\}$ with $q \in \{a, b\}$ such that p and q lie again on opposite sides of the vertical line. For example, in Figure 1a the next position would be $\{p, b\}$. The authors of [6] gave matching upper and lower bounds of order $\log^2(n)$ for the expected duration of this process in the plane.



(a) **(b)**

Figure 1 **(a)** An instance of "one line and n points". **(b)** A possible path that the directed random walk might take on the corresponding polytope.

34th International Symposium on Computational Geometry (SoCG 2018).
Editors: Bettina Speckmann and Csaba D. Tóth; Article No. 60; pp. 60:1–60:14
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The process generalizes naturally to higher dimensions. However, understanding its behavior in any dimension other than 2 has remained a wide open problem, with the notable exception of an $\Omega(\log^3 n)$ lower bound in three dimensions [16]. As the dimension grows, the situation indeed becomes increasingly complicated, and the question has seen no further improvements for the subsequent fifteen years.

The relevance of the generalized process lies in its intimate connection with the Random-Edge simplex algorithm for linear programming, which has already been widely considered before; for example cf. [1, 2, 4, 8, 11]. Random-Edge is naturally formulated in terms of a random walk on the vertex-edge graph of the feasible region (see Figure 1b). To be precise it is a *directed* random walk, because every edge may be used in one prescribed direction only: the direction along which the objective function improves. In each step the directed random walk moves from the current vertex to a vertex chosen uniformly at random from all neighbors with smaller objective value. By means of the so-called *extended Gale transform*, the directed random walk on a d-polytope with n facets translates precisely into the process of "one line and n points" in $\mathbb{R}^r$, where $r := n - d$ denotes the *corank* of the polytope. For lack of space we refer the interested reader to the appendix of [6] for a complete exposition of the extended Gale transform.

The interpretation in terms of polytopes also explains the difficulties that arise when analyzing the process for $r \geq 3$. Namely, it is known that every simple polytope of corank $r = 1$ is a simplex, and every simple polytope of corank $r = 2$ is a product of two simplices; these situations are thus well-understood, classified, and not too complicated. Already for $r = 3$, however, the classification is considerably more involved (cf. chapter 6.3 in [7]), and for $r \geq 4$ no similar classification exists.

The name "corank" is not entirely standard; it has been used e.g. in [14]. Despite its anonymity it plays a prominent role in polytope theory: The *Hirsch conjecture* once stated that the corank might be an upper bound on the diameter of any polytope. Since the Hirsch conjecture, in its strict form, has been disproved by Santos [15], the search for a close connection between these quantities continues. Indeed we believe it fruitful to analyze algorithms for linear programming in a setting where the corank is assumed to be small, i.e., a constant or a slowly growing function of n. A positive result of this type is for example Kalai's Random-Facet algorithm: His upper bound in [10] for the expected number of arithmetic operations becomes polynomial if the corank is taken to be of order $r = O(\log n)$. In contrast, Random-Edge has proved to be notoriously hard to analyze, and tight bounds are rare when we want to understand the behavior of a given simplex algorithm on a complete class of instances. The analysis in [9] for $d = 3$ suggests that Random-Edge can be a good choice in low dimension; the present paper takes the dual viewpoint, fixing the corank rather than the dimension. Note that the mildly exponential lower bound obtained by Friedmann et al. [3] does not pose any restrictions on the corank.

A polytope with n facets and constant corank r has $O(n^r)$ vertices, and this bound is tight; this follows for example from McMullen's theorem [12]. Thus, $O(n^r)$ is a trivial bound for the number of Random-Edge pivoting steps. The known bounds for $r = 2$ suggest that the process outperforms the trivial bound considerably; however, it is not at all clear what can be expected in general. It is conceivable that a bound of $O(\log^r n)$ holds, although we are currently missing the mathematical techniques and insights to prove this. In this paper, we prove that one can at least not do better.

The history of lower-bound constructions for specific LP-solving algorithms shows that it is often considerably easier to prove bounds in an abstract model; this is how *unique sink orientations* (USOs) enter the picture. The same principle applies to the present paper: We

first present a construction in the USO model in Section 2 and strengthen the result to the geometric setting in the rest of the paper. Somewhat atypically, the main ideas underlying the geometric construction are not *entirely* different from the ideas underlying the USO construction.

Our results. We prove that the random process of "one line and n points" in $\mathbb{R}^r$ may take $\Omega(\log^r n)$ steps in expectation. This generalizes previous constructions for the cases $r = 2, 3$ [6, 16]. The exact bound that we obtain is specified in Theorem 13. Using the extended Gale transform, our result can be rephrased in different settings, as in the following theorems. Theorem 1 rephrases our result in the language of linear programs. Theorem 2 uses instead the language of polytopes. The combinatorial type of the polytopes that we construct in this way is rather special: they are *grid polytopes*, i.e., Cartesian products of simplices.

▶ **Theorem 1.** *Let $r \in \mathbb{N}_0$ be a fixed parameter. There are linear programs in d variables with $n = d + r$ constraints on which the* Random-Edge *algorithm needs $\Omega(\log^r n)$ pivoting steps in expectation as $n \to \infty$.*

▶ **Theorem 2.** *Let $r \in \mathbb{N}_0$. There are grid d-polytopes with $n = d + r$ facets on which the directed random walk (with the direction specified by a linear function) needs $\Omega(\log^r n)$ steps in expectation as $n \to \infty$.*

2 Prelude: walks on grids

In this section we prove a lower bound of order $\log^r(n)$ for the expected duration of a random walk on a certain class of directed graphs, namely *unique sink orientations of grids*. The construction does not involve any geometry and should be simpler to understand than the point-set construction in Section 4.

Given a directed (multi-)graph G and a vertex v_0 of G, the *directed random walk* is the random process $v_0, v_1, v_2, \ldots$ described as follows: If the current position is v_i, choose one of the outgoing edges at v_i uniformly at random, and let v_{i+1} be the other endpoint of that edge. The process terminates when (and if) it reaches a sink. The random variable

$$T(G, v_0) = \min\{t : v_t \text{ is a sink}\}$$

will denote the duration of the directed random walk on G starting in v_0. We will abbreviate $T(G) := T(G, v_0)$ for a starting position v_0 chosen uniformly at random from the set of vertices.

Following the terminology in [5], a *grid* is a Cartesian product of a (finite) number of (finite) complete graphs $G_1, \ldots, G_r$. Explicitly, the vertex set of this graph is $V(G_1) \times \cdots \times V(G_r)$, and two vertices $(u_1, \ldots, u_r)$ and $(v_1, \ldots, v_r)$ are joined by an edge if and only if they differ in exactly one coordinate. We could have defined grids equivalently as the vertex-edge graphs of grid polytopes. The number r is usually called the *dimension* of the grid (this is *not* the dimension of the underlying grid polytope!); and its *size* is the number $|V(G_1)|+\cdots+|V(G_r)|$.

A *subgrid* of a grid is an induced subgraph on a set of the form $U_1 \times \cdots \times U_r$ with $U_i \subseteq V(G_i)$. Finally, a *unique sink orientation* of a grid is an orientation of the edges of the grid with the property that every subgrid possesses a unique sink. An example is shown in Figure 2a.

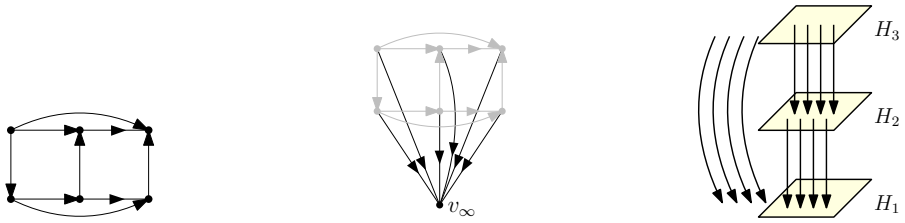


Figure 2 **(a)** A grid G of dimension $r = 2$ and size $n = 5$. **(b)** The augmented (multi-)graph G^Δ, here for $\Delta = 1$. **(c)** A schematic depiction of the grid constructed in the proof of Theorem 3.

▶ **Theorem 3.** *Let $r \in \mathbb{N}_0$ and $m \in \mathbb{N}$. There is an r-dimensional grid G of size $n := rm$, endowed with an acyclic unique sink orientation, such that a directed random walk on the grid, starting at a random position, takes at least*

$$\mathrm{E}\big[T(G)\big] \geq \frac{1}{r!} \ln^r (m+1) - 1 \tag{1}$$

steps in expectation.

We remark that the theorem is meaningful primarily for a fixed dimension r and with the grid size tending to infinity. In this setting it says that the number of pivoting steps can be of order $\Omega(\log^r n)$.

Proof. We can choose the grid to be $G = G_1 \times \cdots \times G_r$ with $G_1 = \cdots = G_r = K_m$. We construct a unique sink orientation on G by induction on r. For $r = 0$, the graph consists of a single vertex, so we need not orient any edges.

For $r \geq 1$, we first choose a permutation of $V(G_r)$ uniformly at random from the set of all permutations, and we label the vertices according to the chosen permutation, as in $V(G_r) = \{1, \ldots, m\}$. Now we partition the grid into "hyperplanes"

$$H_i = G_1 \times \cdots \times G_{r-1} \times \{i\} \quad (i = 1, \ldots, m).$$

Each hyperplane H_i is a subgrid of dimension $r-1$, so we can inductively assign an orientation to it. We do this for each hyperplane independently (i.e., all random permutations used throughout the construction are chosen independently). The only edges that we still need to orient are those between vertices from two distinct hyperplanes. We orient those according to our chosen permutation of $V(G_r)$. Explicitly, the edge from a vertex $(u_1, \ldots, u_{r-1}, i)$ to $(u_1, \ldots, u_{r-1}, j)$ is directed forwards if and only if $i > j$. See Figure 2c for an illustration.

It is easy to verify that this defines an acyclic unique sink orientation of the grid. We will now analyze the duration of the random walk from a random starting position. Concerning the starting position, here is a key observation: Due to the random permutations involved in the construction of the grid orientation, it amounts to the same random process whether we start the walk in a random position of the grid, or whether we start in any fixed given position. Consequently, if the random walk visits one of the hyperplanes H_i, we can relate the behavior within H_i to the $(r-1)$-dimensional construction, for which we have a lower bound by induction. However, there *is* a difference between H_i and the $(r-1)$-dimensional construction: namely, every vertex of H_i has $i-1$ additional outgoing edges by which the walk may leave H_i at any moment. To account for these, we make use of the following "augmented" multigraph; see Figure 2b for an example.

▶ **Definition 4.** Given any (multi-)graph Γ and a parameter $\Delta \in \mathbb{N}_0$, we define an *augmented multigraph* Γ^Δ as follows. We add a new, special, vertex v_∞ to the vertex set of Γ. Furthermore we add Δ many edges $\overrightarrow{vv_\infty}$ for every $v \in V(\Gamma)$. If $\Delta = 0$ then we add one additional edge $\overrightarrow{sv_\infty}$ for every sink s of Γ; this way we ensure that v_∞ is the only sink of Γ^Δ.

▶ **Lemma 5.** *Let $\Delta \in \mathbb{N}_0$. Then the expected duration of the directed random walk on the augmented construction G^Δ, starting from a random position (and ending in v_∞), satisfies the bound*

$$\mathrm{E}\big[T(G^\Delta)\big] \geq \frac{1}{r!}\left(\ln\left(m+\Delta+1\right) - \ln(\Delta+1)\right)^r.$$

Proof of the lemma. We proceed by induction on $r \geq 0$. If $r = 0$ then G consists of a single vertex and there is nothing to prove; so let $r \geq 1$. For $i \in \{1,\dots,m\}$, let T_i denote the number of positions that the walk visits on H_i, so that the total duration of the walk is given by

$$T(G^\Delta) = T_1 + \cdots + T_m. \tag{2}$$

Furthermore, let $\mathcal{E}_i$ denote the event that the random walk visits at least one vertex in H_i $(i = 1,\dots,m)$. We claim

$$\Pr\big[\mathcal{E}_i\big] \geq \frac{1}{\Delta+i}. \tag{3}$$

To this end we consider the hitting time

$$\tau_i := \min\big\{t \;:\; v_t \in \{v_\infty\} \cup H_1 \cup \cdots \cup H_i\big\}$$

where, as before, $v_t \in V(G^\Delta)$ denotes the position that the random walk visits at time t $(t = 0, 1, 2, \dots)$. Note that the hyperplanes H_i are visited in decreasing order; so either the walk visits H_i at time τ_i, or not at all. Hence, $\mathcal{E}_i$ equals the event $\{v_{\tau_i} \in H_i\}$. We now calculate the probability of this event by conditioning on $\tau_i \geq 1$.

Case 1: $\tau_i \geq 1$. Since $v_{\tau_i - 1}$ has Δ outgoing edges to v_∞ and one outgoing edge to each of the hyperplanes $H_1,\dots,H_i$, and since the random walk is equally likely to move along any of these $\Delta + i$ edges, we obtain

$$\Pr\big[\mathcal{E}_i \,\big|\, \tau_i \geq 1\big] = \Pr\big[v_{\tau_i} \in H_i \,\big|\, \tau_i \geq 1\big] = \frac{1}{\Delta+i}. \tag{4}$$

Case 2: $\tau_i = 0$. Here we need to look at v_0, which (conditioned on $\tau_i = 0$) is a vertex taken uniformly at random from the set $H_1 \cup \cdots \cup H_i$. Since the hyperplanes $H_1,\dots,H_i$ are all of equal cardinality, we obtain

$$\Pr\big[\mathcal{E}_i \,\big|\, \tau_i = 0\big] = \Pr\big[v_0 \in H_i \,\big|\, \tau_i = 0\big] = \frac{1}{i} \geq \frac{1}{\Delta+i}. \tag{5}$$

The claim (3) follows by combining (4) and (5). Now, it is easy to see that $T_i|\mathcal{E}_i$ has the same distribution as $T((H_i)^{\Delta+i-1})$, so that we obtain

$$\begin{aligned}\mathrm{E}\big[T_i\big] &= \Pr\big[\mathcal{E}_i\big] \cdot \mathrm{E}\big[T_i \,\big|\, \mathcal{E}_i\big] \\ &\geq \frac{1}{\Delta+i} \cdot \mathrm{E}\big[T((H_i)^{\Delta+i-1})\big]\end{aligned}$$

$$\geq \frac{1}{\Delta+i} \cdot \frac{1}{(r-1)!} \big(\ln(m+\Delta+i) - \ln(\Delta+i)\big)^{r-1}$$

where the last step was using the induction hypothesis. With (2) we obtain

$$\begin{aligned}
\mathrm{E}[T] &= \sum_{i=1}^{m} \mathrm{E}[T_i] \\
&\geq \sum_{i=1}^{m} \frac{1}{\Delta+i} \cdot \frac{1}{(r-1)!} \big(\ln(m+\Delta+i) - \ln(\Delta+i)\big)^{r-1} \\
&\geq \sum_{i=1}^{m} \frac{1}{\Delta+i} \cdot \frac{1}{(r-1)!} \big(\ln(m+\Delta+1) - \ln(\Delta+i)\big)^{r-1} \\
&\geq \int_{1}^{m+1} \frac{1}{\Delta+x} \cdot \frac{1}{(r-1)!} \big(\ln(m+\Delta+1) - \ln(\Delta+x)\big)^{r-1} \,\mathrm{d}x \\
&= \Big[-\frac{1}{r!} \big(\ln(m+\Delta+1) - \ln(\Delta+x)\big)^{r} \Big]_{x=1}^{m+1} \\
&= \frac{1}{r!} \big(\ln(m+\Delta+1) - \ln(\Delta+1)\big)^{r}
\end{aligned}$$

which concludes the proof of the lemma. In order to deduce the theorem, we choose $\Delta = 0$ to obtain a random orientation (i.e., a probability distribution of orientations) of G^0 such that

$$\mathrm{E}[T(G^0)] \geq \frac{1}{r!} \ln^r(m+1). \tag{6}$$

A directed random walk on G corresponds to a random walk on G^0, except that the latter does one additional step in the end (from the sink of G to the extra vertex v_∞). Thus we need to subtract 1 from the bound (6) to obtain the desired bound (1). We are left only to observe that there must then also exist at least one concrete (not random) choice of orientation G that satisfies this bound. This concludes the proof of Theorem 3. ◀

In Theorem 3 we chose n to be a multiple of r. For other values of n we can still deduce essentially the same bound, as in the following corollary.

▶ **Corollary 6.** *For all $r, n \in \mathbb{N}$ with $n > r$ there is an r-dimensional acyclic unique sink orientation of a grid $\hat{G}$ of size n such that $\mathrm{E}[T(\hat{G})] > \frac{1}{r!} \ln^r\left(\frac{n}{r}\right) - 1$.*

Proof. If n is divisible by r, then the corollary is an immediate consequence of Theorem 3. Assume that n is not divisble by r, and let $m = \lfloor \frac{n}{r} \rfloor$. We take our construction G from the proof of Theorem 3 for the size $n' := rm$ and embed it into a grid $\hat{G}$ of size n. We can do this in such a fashion that all the edges between G and $\hat{G} \setminus G$ point into G. In this way we obtain an acyclic unique sink orientation on $\hat{G}$ with $\mathrm{E}[T(\hat{G})] \geq \mathrm{E}[T(G)]$, which yields the corollary using (1) with $m > \frac{n}{r} - 1$. ◀

3 One line and n points

Here we describe the geometric setting in which we prove our main theorem. We will assume that we are given a set of n points $A \subseteq \mathbb{R}^r$ and a non-zero vector $u \in \mathbb{R}^r$. Its linear span $\mathbb{R}u$ is a line: the "one line" or *requirement line* featured in the heading of this section.

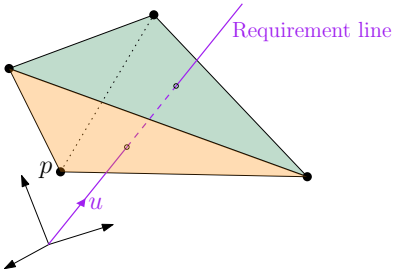


Figure 3 A tetrahedron in $\mathbb{R}^3$. Its two front facets, the green and orange triangles, are both a pierced simplex. If S denotes the green triangle at the top, then the *simplex obtained by pivoting at S with p* is the orange triangle at the bottom.

Pierced simplices and general position. We call a set $S \subseteq A$ *pierced* or, more exactly, *u-pierced* if the convex hull $\mathrm{conv}(S)$ intersects the requirement line $\mathbb{R}u$. If in addition $|S| = r$, then S is a *pierced simplex*. Some readers might find it more natural to reserve the term "simplex" for the set $\mathrm{conv}(S)$ instead of S; we will always take care to distinguish between the two whenever the distinction is important. A pierced simplex S is *non-degenerate* if (I) S is affinely independent, (II) no proper subset of S is a pierced set, and (III) the affine hyperplane spanned by S and the requirement line are not parallel. — Within this paper, when we say that (A, u) is *in general position*, we merely mean that every u-pierced simplex $S \subseteq A$ is non-degenerate.

Below and above. Consider the affine span of a non-degenerate pierced simplex $S \subseteq A$: it is a hyperplane in $\mathbb{R}^r$ that is not parallel to the vector u. The direction of u thus determines an orientation (a positive and negative side) of this hyperplane. For any point $x \in A$, we will say that x lies *(strictly) below* S if it lies on the (strictly) negative side of the hyperplane. The word "above" is understood similarly.

Pivoting steps. Given (A, u) in general position, a pierced simplex $S \subseteq A$ and a point $p \in A$ strictly below S, we define a new pierced simplex $S' \subseteq A$ which we call the *simplex obtained by pivoting at S with p*. To this end consider the set $S \cup \{p\}$: It is an r-dimensional simplex, and the boundary of $\mathrm{conv}(S \cup \{p\})$ is pierced by the line $\mathbb{R}u$ exactly twice: once in the facet S, and once in another facet which we take to be S'. See Figure 3 for an example. We note that

- S' is by general position uniquely determined,
- S' is a subset of $S \cup \{p\} \subseteq A$, and
- S' is a pierced simplex.

The random process. Given a finite set $A \subseteq \mathbb{R}^r$ of n points, a non-zero vector $u \in \mathbb{R}^r$, and a u-pierced simplex $S_0 \subseteq A$, we define the following random process, denoted $\mathcal{R}(A, u, S_0)$. We will keep on assuming that (A, u) is in general position.

The states (or positions) of the process are pierced simplices, and S_0 is the starting position. The consecutive positions $S_1, S_2, \ldots$ are obtained as follows. If the current position is S_i, let p_i be a point (the ith "pivot") chosen uniformly at random from the set of points from A that lie strictly below S_i. (If there are no such points, then the random process terminates at this stage.) Now define S_{i+1} to be the simplex obtained by pivoting at S_i with p_i. — Our main theorem in Section 5 states that the expected number of steps until the process terminates can be of order $\log^r(n)$.

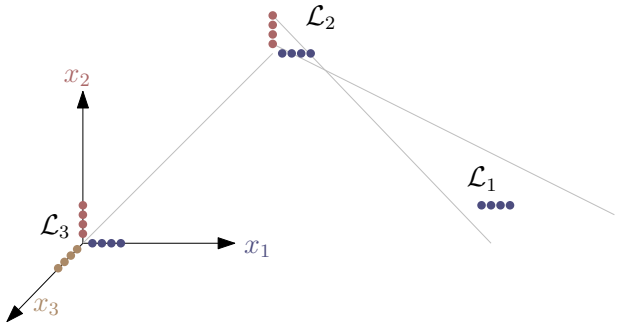


Figure 4 The constructed point set for $r = 3$, $m = 4$. The sketch is not true to scale. All off-axis points lie in the plane $x_3 = -64$.

4 Construction

Here we construct the point set that underlies the proof of our main theorem (Theorem 13 in Section 5).

Points, colors, layers, and phases. For all $r, m \in \mathbb{N}$ we define our point set $A(r, m) \subseteq \mathbb{R}^r$ as follows; a sketch is shown in Figure 4. We use the notation $\mathbf{0}_r = (0, \ldots, 0)$ for the all-zeros vector in $\mathbb{R}^r$. We let

$$A(r, m) := \{\, a_{i,j,k} \,:\, i, j \in [r],\ i \le j,\ k \in [m] \,\}$$

where

$$a_{i,j,k} := \begin{pmatrix} \mathbf{0}_{i-1} \\ \left(m^3 + m^5 + \cdots + m^{2r-2j+1}\right) + (r-j)m + k \\ \mathbf{0}_{j-i} \\ -m^{2r-2j+1} \\ -m^{2r-2j-1} \\ \vdots \\ -m^5 \\ -m^3 \end{pmatrix}.$$

In particular, for $j = r$, we have $a_{i,r,k} = k\mathbf{e}_i$, where $\mathbf{e}_i$ denotes the ith standard unit vector in $\mathbb{R}^r$. We call the indices i, j, k the *color*, the *layer*, and the *phase* of a point, respectively. Sometimes we will need a notational shorthand for colors and layers, so we define $\mathcal{C}_i \subseteq A(r, m)$ to denote the set of points of color i, and $\mathcal{L}_j \subseteq A(r, m)$ to denote the set of points from layer j. So defined, $\mathcal{L}_j$ consists of jm points, and our point set consists of $n = \binom{r+1}{2} \cdot m$ points overall.

We will fix u to denote the all-ones vector, $u = \mathbf{1}_r$, so that a set $S \subseteq A(r, m)$ is pierced if and only if its convex hull intersects the diagonal line $\mathbb{R}\mathbf{1}_r$. The rest of this section is devoted to a number of lemmas concerning the relevant structure of our construction. We begin by identifying the pierced subsets.

▶ **Lemma 7.** *Let $S \subseteq A(r, m)$ be a pierced subset. Then S contains a point of color i, for all $i \in [r]$.*

The next lemma shows the converse of Lemma 7, implying that the set of pierced simplices can be identified with the set $\mathcal{C}_1 \times \cdots \times \mathcal{C}_r$. More to the point, this implies that the extended Gale transform of our point set defines a grid polytope.

▶ **Lemma 8.** *Let $S \subseteq A(r,m)$ and assume that S contains a point from each color class $\mathcal{C}_1, \ldots, \mathcal{C}_r$. Then S is a pierced subset. Furthermore, for each $i \in [r]$, $\operatorname{conv}(S)$ intersects the ith coordinate axis in some point $t\mathbf{e}_i$ with $t > 0$.*

▶ **Lemma 9.** *$(A, \mathbf{1}_r)$ is in general position; that is, every pierced simplex $S \subseteq A(r,m)$ is non-degenerate.*

Lemma 9 above assures that the random process associated with our construction is well-defined. The next lemma states that, as our random process evolves, the intersection value t of the current position with the ith coordinate axis ($i = 1, \ldots, r$) is monotonically decreasing with time and, thus, can serve as a measure of progress. This is of course by no means true for an arbitrary point set, but it holds in the case of our construction. Next, Lemma 11 states a simple condition from which to tell whether the points from the layer $\mathcal{L}_{r-1}$ lie above or below the current position; this condition is immediately relevant for the analysis of the random process.

▶ **Lemma 10.** *Let $i \in [r]$. Let $S \subseteq A(r,m)$ be a pierced simplex, let $p \in A$ be a point strictly below S, and let S' denote the pierced simplex obtained by pivoting at S with p. Let $t_i\mathbf{e}_i$ and $t'_i\mathbf{e}_i$ denote the intersection of $\operatorname{conv}(S)$ (respectively, $\operatorname{conv}(S')$) with the ith coordinate axis, as in Lemma 8. Then we have $t'_i \leq t_i$ for all i.*

▶ **Lemma 11.** *Let $r \geq 2$, $m \geq 2$, let $S \subseteq A(r,m)$ be a pierced simplex, and let $t_i\mathbf{e}_i$ denote the intersections of $\operatorname{conv}(S)$ with the ith coordinate axis as in Lemma 8 ($i = 1, \ldots, r$).*

(a) *If, for some i, $t_i \leq t_r$, then all points from $\mathcal{C}_i \cap \mathcal{L}_{r-1}$ lie strictly* above *S.*

(b) *If, for some i, $t_i \geq t_r + 1$, then all points from $\mathcal{C}_i \cap \mathcal{L}_{r-1}$ lie strictly* below *S.*

(c) *If i is such that $t_i = \min\{t_1, \ldots, t_r\}$, then all points from $\mathcal{C}_i \setminus \mathcal{L}_r$ lie strictly* above *S. In particular we then have $t_i\mathbf{e}_i \in S$.*

The last lemma in this section states that taking the point set $A(r+1,m)$ and removing the outermost layer $\mathcal{L}_{r+1}$ yields a point set that is equivalent, for our purposes, to the set $A(r,m)$. This observation is key to the inductive approach followed in section 5. Actually, the statement is a bit more general: The lemma starts from the set $A(R,m)$ for any $R > r$ and then removes all higher layers $\mathcal{L}_{r+1}, \ldots, \mathcal{L}_R$.

▶ **Lemma 12.** *Assume $m \geq 3$. For $R > r$, let*

$$B := \{(x_1, \cdots, x_r) : x \in A(R,m),\ x \in \mathcal{L}_1 \cup \cdots \cup \mathcal{L}_r\}.$$

Then Lemmas 7 to 11 are also valid for the point set B in place of $A(r,m)$.

5 Analysis

The goal of this section is to prove the main theorem of this paper, concerned with the random process $\mathcal{R}_{r,m} = \mathcal{R}(A, u, S_0)$ associated with the point set $A := A(r,m)$, the all-ones vector $u := \mathbf{1}_r$, and the starting position $S_0 := \{m\mathbf{e}_1, \ldots, m\mathbf{e}_r\}$:

▶ **Theorem 13.** *The expected number of steps performed by the random process $\mathcal{R}_{r,m}$ is at least*

$$\frac{1}{r!^3}\left(\ln\left(m + \tbinom{r}{2} + \Delta\right) - \ln\left(1 + \tbinom{r}{2} + \Delta\right)\right)^r = \Omega(\log^r m).$$

In terms of the number of points, n, the bound can be written in the form $\Omega(\log^r n)$.

The augmented process $\mathcal{R}^{\Delta}_{r,m}$. In order to make an inductive proof possible, we will make use of an "augmented" pivoting process, in analogy to the "augmented graph" that we used in the proof of Theorem 3. Given a number $\Delta \geq 0$, the *augmented process* $\mathcal{R}^{\Delta}_{r,m}$ is defined as follows.

- The starting position is chosen by an adversary, in the following way. The adversary chooses one new point $\alpha_i \mathbf{e}_i$ on each axis, subject to the constraint $\alpha_i \geq m$. These points are added to the point set $A = A(r,m)$ to obtain an augmented point set

  $$A' := A \cup \{\alpha_i \mathbf{e}_i \,:\, i \in [r]\},$$

  and the starting position is now chosen as $S_0 := \{\alpha_1 \mathbf{e}_1, \ldots, \alpha_r \mathbf{e}_r\}$. We fit the new points into our terminology of colors, layers and phases by saying that $\alpha_i \mathbf{e}_i$ has color i, layer r and phase $m+1$; and we remark that the lemmas in Section 3 still hold for the augmented point set.
- The positions $S_0, S_1, \ldots$ of the augmented process are pierced simplices of A', except that we also introduce a new, special, position S_∞. (S_∞ is just a formal symbol; it is not represented by any simplex.) This will be the terminal position.
- If we are currently at position S_t, then the next position S_{t+1} is obtained as follows: Let $\mathrm{below}(S_t) \subseteq A'$ denote the set of points that lie strictly below S_t. We draw a pivot element p_{t+1} from the set $\mathrm{below}(S_t) \cup \{\infty\}$ according to the distribution

  $$\Pr\big[p_{t+1} = x\big] = \begin{cases} \frac{1}{|\mathrm{below}(S_t)|+\Delta} & \text{for } x \in \mathrm{below}(S_t), \\ \frac{\Delta}{|\mathrm{below}(S_t)|+\Delta} & \text{for } x = \infty. \end{cases}$$

  If $p_{t+1} = \infty$, then $S_{t+1} = S_\infty$, and the process terminates. Otherwise we perform a standard pivoting step at S_t with p_{t+1}. (Edge case: If $\Delta = 0$ and $\mathrm{below}(S_t) = \emptyset$, then we always pick $p_{t+1} = \infty$.)

Note that, despite its name, the "augmented" process typically terminates earlier than the non-augmented process: the larger the parameter Δ is, the sooner! For $\Delta = 0$ the augmented process behaves like the original, non-augmented process — except for the modified starting position and one additional final pivoting step towards the terminal position S_∞.

The phase of a pierced simplex. We define the *phase* of a pierced simplex $S \subseteq A'$ as

$$\mathrm{phase}(S) := \min\{\mathrm{phase}(p) \,:\, p \in S \text{ with } \mathrm{layer}(p) = r\}.$$

The minimum is well-defined because S contains at least one point from the layer $\mathcal{L}_r$: Indeed Lemma 7 tells us that S contains a point from $\mathcal{C}_r$, which is a subset of $\mathcal{L}_r$. For consistency we also define $\mathrm{phase}(S_\infty) := 0$. — Note that we are overloading the term "phase", because we have defined the phase of a point earlier.

We remark that the phase of a pierced simplex can be equivalently written as $\mathrm{phase}(S) = \min\{t_1, \ldots, t_r\}$, where $t_i \mathbf{e}_i$ denotes the intersection of $\mathrm{conv}(S)$ with the ith coordinate axis; this follows from Lemma 11(c). Using this observation, the possible choices for pivots at the time of a phase change are easily identified, and from this information we may read off the probability that a particular phase is visited. The following lemma summarizes the result of this observation.

▶ **Lemma 14** (The phases visited by the augmented process). *Let $\sigma_1 < \sigma_2 < \cdots < \sigma_N$ denote the times at which a phase change occurs in the augmented random process, and let $(\phi_i)_{0 \leq i \leq N}$ denote the phases that are visited, i.e., $\phi_0 = m+1$, and $\phi_i = \mathrm{phase}(S_{\sigma_i})$ $(1 \leq i \leq N)$. Then we have, for $i \geq 1$:*

(a) *The distribution of ϕ_i is given by*

$$\Pr\big[\phi_i = x \,\big|\, \phi_{i-1}\big] = \begin{cases} \frac{r}{r(\phi_{i-1}-1)+\Delta} & \text{for } x \in [\phi_{i-1}-1], \\ \frac{\Delta}{r(\phi_{i-1}-1)+\Delta} & \text{for } x = 0. \end{cases}$$

(b) *If $\phi_i > 0$, then the color of the pivot at time σ_i is a u.a.r. element of $[r]$.*

Consider one of the phases ϕ_i that are visited by the augmented random process. We want to bound the *duration* of the phase ϕ_i, i.e. the number $\sigma_{i+1} - \sigma_i$, from below. In general this duration could be very short, so we introduce a suitable notion of a *good phase*. The definition will guarantee that, when entering a good phase, all points of the layer $\mathcal{L}_{r-1}$ will lie strictly below the current position; and this property will in turn make it possible to derive a lower bound on the duration of a good phase inductively.

▶ **Definition 15** (good phases)**.** Let $k \in [m]$. We say that k is a *good phase* of the augmented process if, using the notation from Lemma 14,

(i) k is visited, so that $k = \phi_j$ for some j,
(ii) the pivot at time σ_j has color r, and
(iii) the position S_{σ_j} does not contain any point from the layer $\mathcal{L}_{r-1}$.

Let the reader be warned that the above definition is weaker than one might think at first: The only points that we take directly into account are those from the two outermost layers $\mathcal{L}_r$ and $\mathcal{L}_{r-1}$. In particular we allow S_{σ_j} to contain points from other layers. When reading on, it is useful to keep in mind one consequence of Lemma 11(c): The phase of the current position can change only when pivoting a point from the layer $\mathcal{L}_r$, i.e., a point that lies on one of the coordinate axes. Consequently, pivots in lower layers can largely be ignored in our analysis.

▶ **Lemma 16.** *Let $1 \le k \le m-1$. Then phase k is a good phase with probability at least $\frac{1}{r(\Delta+kr)}$.*

Proof. Let j be the (random) largest index such that $\phi_{j-1} > k$. Then the probability of (i) equals $\Pr\big[\phi_j = k\big]$, and using Lemma 14(a) we compute this probability to be $\frac{r}{rk+\Delta}$. Given (i), Lemma 14(b) tells us that the probability of (ii) equals $1/r$.

Assume that (i) and (ii) hold. It remains to show that, in this case, (iii) holds with probability at least $1/r$. We consider the position S_{σ_j-1} one time step before entering phase k. Using the same notation as in Lemma 11, let $t_i\mathbf{e}_i$ denote the intersections of $\operatorname{conv}(S_{\sigma_j-1})$ with the ith coordinate axis ($i = 1, \dots, r$). Note that $t_i > k$ for all i, because phase k has not been entered yet at this time.

By Lemma 11 it is sufficient to give a bound for the event

$$t_i \le t_r \text{ for all } i = 1, \dots, r-1. \tag{7}$$

To this end, let τ denote the time that the point $t_r\mathbf{e}_r$ is pivoted, so that S_τ is the first position to include the point $t_r\mathbf{e}_r$. Note that t_r does not change in between time τ and time σ_j; thus, if property (7) already holds at time τ, then by monotonicity (Lemma 10) it will still hold at time S_{σ_j-1}. So assume that at time τ property (7) does not yet hold, so that there are some "bad" indices i with $t_i > t_r$. Let $I \subseteq [r-1]$ denote the set of such bad indices, and let $\tau_1 > \tau$ be the first time that another point of layer r with phase $\le t_r$ and color contained in $I \cup \{r\}$ is pivoted. With probability at least $\frac{|I|}{|I|+1}$, the color of this pivot

is contained in I, in which case the number of bad indices is reduced by 1. Iterating this argument, the number of bad indices will be reduced down to zero with probability at least

$$\frac{|I|}{|I|+1} \cdot \frac{|I|-1}{|I|} \cdots \frac{1}{2} = \frac{1}{|I|+1} \geq \frac{1}{r},$$

as desired. ◀

When the augmented process enters a good phase k, then all the points of the layer $\mathcal{L}_{r-1}$ lie strictly below the current position. We restrict our attention to the hyperplane $x_r = -m^3$ that contains all the points from $A(r,m) \setminus \mathcal{L}_r$: Due to Lemma 12, the augmented process within this hyperplane behaves like the augmented process on the lower-dimensional construction $A(r-1,m)$. However, the process might at any point pivot one of the points $\{p \in \mathcal{L}_r : \text{phase}(p) < k\}$, and as soon as this happens, the good phase k already ends. We can account for this by adjusting the parameter Δ and we obtain the following lemma.

▶ **Lemma 17.** *For every $1 \leq k \leq m-1$, if phase k is visited and if it is a good phase, then its expected duration is bounded from below by the best-case*[1] *expected duration of the process $\mathcal{R}^{\Delta+(k-1)r}_{r-1,m}$.*

▶ **Theorem 18.** *Let $t^{\Delta}_{r,m}$ denote the best-case expected duration of the augmented process $\mathcal{R}^{\Delta}_{r,m}$. Then we have*

$$t^{\Delta}_{r,m} \geq \frac{1}{r!^3} \cdot \Big(\ln(m + \tbinom{r}{2} + \Delta) - \ln(1 + \tbinom{r}{2} + \Delta)\Big)^r.$$

Proof. By induction on r. For $r = 1$ the statement is easy to verify; let now $r \geq 2$. Combining Lemmas 16 and 17 we obtain

$$t^{\Delta}_{r,m} \geq \sum_{k=1}^{m-1} \frac{1}{r(kr+\Delta)} \cdot t^{\Delta+kr-1}_{r-1,m}. \tag{8}$$

The induction hypothesis gives, for $1 \leq k \leq m-1$,

$$\begin{aligned} t^{\Delta+kr-1}_{r-1,m} &\geq \frac{1}{(r-1)!^3} \cdot \Big(\ln(m + \tbinom{r-1}{2} + \Delta + kr - 1) - \ln(\tbinom{r-1}{2} + \Delta + kr)\Big)^{r-1}. \\ &= \frac{1}{(r-1)!^3} \cdot \Big(\ln(m + \tbinom{r}{2} + \Delta + kr - r) - \ln(\underbrace{1 + \tbinom{r}{2} + \Delta + kr - r}_{=:f(k)})\Big)^{r-1}, \end{aligned} \tag{9}$$

where we have used Pascal's rule to handle the binomial coefficients. Plugging (9) into (8) and furthermore using the simple inequality $kr + \Delta \leq f(k)$, we thus obtain

$$\begin{aligned} t^{\Delta}_{r,m} &\geq \frac{1}{(r-1)!^3} \sum_{k=1}^{m-1} \frac{1}{rf(k)} t^{\Delta+kr-1}_{r-1,m} \\ &\geq \frac{1}{(r-1)!^3} \int_{x=1}^{m} \frac{1}{rf(x)} \Big(\ln(m + \tbinom{r}{2} + \Delta + rx - r) - \ln f(x)\Big)^{r-1} \,\mathrm{d}x \qquad (10) \\ &\geq \frac{1}{(r-1)!^3} \int_{x=1}^{1+(m-1)/r} \frac{1}{rf(x)} \Big(\ln(m + \tbinom{r}{2} + \Delta) - \ln f(x)\Big)^{r-1} \,\mathrm{d}x \qquad (11) \end{aligned}$$

[1] The term "best-case" here, as well as in Theorem 18, refers to the action of the adversary who chooses the starting position of the augmented process. The intended meaning is for the lower bound in Theorem 18 to hold for any choice of starting position.

$$= \frac{1}{r!^3} \cdot \Big[-\Big(\ln(m + \tbinom{r}{2} + \Delta) - \ln f(x)\Big)^r \Big]_{x=1}^{1+(m-1)/r}$$
$$= \frac{1}{r!^3} \cdot \Big(\ln(m + \tbinom{r}{2} + \Delta) - \ln(1 + \tbinom{r}{2} + \Delta)\Big)^r,$$

which proves the theorem. Note that the integrand in (10) is positive everywhere, so we were justified to restrict the range of the integral in (11), effectively dropping negligible terms.

Theorem 13 now follows from Theorem 18 by setting $\Delta = 0$. ◀

6 Conclusion

Outlook. It remains an open question whether one can obtain good upper bounds for the expected number of steps performed by Random-Edge when the corank is bounded. The only non-trivial result at this point remains the $O(\log^2 n)$ bound for the case $r = 2$, which was settled in [6] and which the author has studied further in a more abstract setting in [13]. It might well be that there is a threshold behavior in the sense that Random-Edge performs well for slowly growing r, and badly for quickly growing r. Finally, the same questions can be asked for other simplex pivoting rules that might be easier to analyze.

Remark on the dependence on r. The leading factor $1/r!^3$ in Theorem 18 is rather small. Due to the results by Friedmann et al. [3], this factor cannot in general be tight. Unfortunately, an improvement seems to be beyond the scope of our method. For the interpretation of Theorem 2 some readers may find it interesting to pick a value of r that depends on the number of facets n. Not every such choice of r leads to a meaningful bound; but it is possible to choose $r = r(n) = \ln^{1/s}\left(1 + 4n/r^4\right)$ with any $s > 3$, which leads to a lower bound of the form $(\ln n)^{\Omega(\ln^{1/s} n)}$ for the directed random walk on a grid polytope with n facets and corank $r(n)$.

References

1 Andrei Z. Broder, Martin E. Dyer, Alan M. Frieze, Prabhakar Raghavan, and Eli Upfal. The worst-case running time of the random simplex algorithm is exponential in the height. *Inf. Process. Lett.*, 56(2):79–81, 1995.

2 Stefan Felsner, Bernd Gärtner, and Falk Tschirschnitz. Grid orientations, $(d, d+2)$-polytopes, and arrangements of pseudolines. *Discrete & Computational Geometry*, 34(3):411–437, 2005.

3 Oliver Friedmann, Thomas Dueholm Hansen, and Uri Zwick. Subexponential lower bounds for randomized pivoting rules for the simplex algorithm. In *Proceedings of the 43rd ACM Symposium on Theory of Computing (STOC)*, pages 283–292, 2011.

4 Bernd Gärtner, Martin Henk, and Günter M. Ziegler. Randomized simplex algorithms on klee-minty cubes. *Combinatorica*, 18(3):349–372, 1998.

5 Bernd Gärtner, Walter D. Morris, Jr., and Leo Rüst. Unique sink orientations of grids. *Algorithmica*, 51:200–235, 2008.

6 Bernd Gärtner, József Solymosi, Falk Tschirschnitz, Pavel Valtr, and Emo Welzl. One line and n points. *Random Structures & Algorithms*, 23(4):453–471, 2003 (preliminary version at STOC 2001).

7 Branko Grünbaum. *Convex polytopes*. Springer, New York, 1967/2003.

8 Kathy Williamson Hoke. Completely unimodal numberings of a simple polytope. *Discrete Applied Mathematics*, 20(1):69–81, 1988.

9 Volker Kaibel, Rafael Mechtel, Micha Sharir, and Günter M. Ziegler. The simplex algorithm in dimension three. *SIAM Journal on Computing*, 34(2):475–497, 2005.

10 Gil Kalai. A subexponential randomized simplex algorithm. In *Proc. 24th ACM Symposium on Theory of Computing*, pages 475–482, 1992.

11 Jiří Matoušek and Tibor Szabó. Random edge can be exponential on abstract cubes. *Advances in Mathematics*, 204(1):262–277, 2006.

12 Peter McMullen. On the upper-bound conjecture for convex polytopes. *Journal of Combinatorial Theory, Series B*, 10(3):187–200, 1971.

13 Malte Milatz. Directed random walks on polytopes with few facets. *Electronic Notes in Discrete Mathematics*, 61:869–875, 2017. The European Conference on Combinatorics, Graph Theory and Applications (EUROCOMB '17).

14 Julian Pfeifle and Günter M. Ziegler. On the monotone upper bound problem. *Experimental Mathematics*, 13(1):1–12, 2004.

15 Francisco Santos. A counterexample to the Hirsch Conjecture. *Annals of Mathematics*, 176:383–412, 2012.

16 Falk Tschirschnitz. *LP-related properties of polytopes with few facets.* PhD thesis, ETH Zürich, 2003.

Table Based Detection of Degenerate Predicates in Free Space Construction

Victor Milenkovic[1]
Department of Computer Science, University of Miami
Coral Gables, FL 33124-4245, USA
vjm@cs.miami.edu

Elisha Sacks[2]
Computer Science Department, Purdue University
West Lafayette, IN 47907-2066, USA
eps@purdue.edu

Nabeel Butt
Facebook
1 Hacker Way, Menlo Park, CA 94025, USA
nfb@fb.com

Abstract

The key to a robust and efficient implementation of a computational geometry algorithm is an efficient algorithm for detecting degenerate predicates. We study degeneracy detection in constructing the free space of a polyhedron that rotates around a fixed axis and translates freely relative to another polyhedron. The structure of the free space is determined by the signs of univariate polynomials, called angle polynomials, whose coefficients are polynomials in the coordinates of the vertices of the polyhedra. Every predicate is expressible as the sign of an angle polynomial f evaluated at a zero t of an angle polynomial g. A predicate is degenerate (the sign is zero) when t is a zero of a common factor of f and g. We present an efficient degeneracy detection algorithm based on a one-time factoring of every possible angle polynomial. Our algorithm is 3500 times faster than the standard algorithm based on greatest common divisor computation. It reduces the share of degeneracy detection in our free space computations from 90% to 0.5% of the running time.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases free space construction, degenerate predicates, robustness

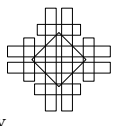
Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.61

Related Version A full version of this paper is available at `http://arxiv.org/abs/1803.06908`.

1 Introduction

An implementation of a computational geometry algorithm is robust if for every input the combinatorial output is correct and the numerical output is accurate. The challenge is to implement the predicates in the algorithms: the signs of algebraic expressions whose variables are input parameters. A predicate is *degenerate* if its sign is zero. A nondegenerate predicate can usually be evaluated quickly, using machine arithmetic. However, detecting that a predicate is degenerate requires more costly computation.

[1] Supported by NSF CCF-1526335.
[2] Sacks and Butt supported by NSF CCF-1524455.



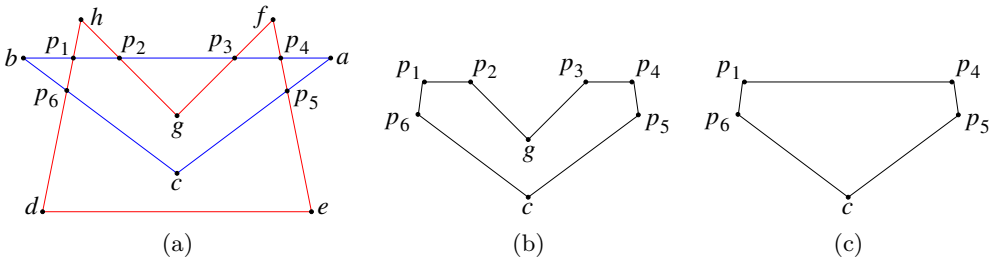


Figure 1 (a) Polygons, (b) intersection, and (c) convex hull.

We present research in degeneracy detection. Prior research mainly addresses degeneracy due to nongeneric input, such as the signed area of a triangle with collinear vertices. Such degeneracy is easily eliminated by input perturbation [3]. We address predicates, which we call *identities,* that are degenerate for all choices of the input parameters. One example is the signed area of a triangle pab with p the intersection point of line segments ab and cd, which is identical to zero when p is replaced by its definition in terms of the input. This identity occurs when constructing the convex hull of the intersection of two polygons. Figure 1 shows generic polygons abc and $defgh$ that intersect at points $\{p_1, \ldots, p_6\}$. The convex hull algorithm encounters an identity when it evaluates the signed area of any three of $\{p_1, p_2, p_3, p_4\}$. Triangulating the intersection of two polygons involves similar identities.

Identities are common when the output of one algorithm is the input to another. The second algorithm evaluates polynomials (the signed area in our example) on arguments (the p_i in our example) that are derived from input parameters by the first algorithm. When these algebraic expressions are rational, the identities are amenable to polynomial identity detection [9]. We are interested in identities involving more general algebraic expressions.

There are two general approaches to identity detection (Sec. 2). One [1] uses software integer arithmetic, computer algebra, and root separation bounds to detect all degenerate predicates, including identities. The second adds identity detection logic to computational geometry algorithms. In the convex hull example, this logic checks if three points lie on a single input segment. The first approach can greatly increase the running time of the software and the second approach can greatly increase the software development time.

We present a new approach to identity detection that avoids the high running time of numerical identity detection and the long development time of identity detection logic. The approach applies to a class of computational geometry algorithms with a common set of predicates. We write a program that enumerates and analyzes the predicates. The predicates are represented by algebraic expressions with canonical variables. The hull example requires 24 canonical variables for the coordinates of the at most 12 input points that define the three arguments of the signed area. When implementing the algorithms, we match their arguments against the canonical variables and use the stored analysis to detect identities.

We apply this approach in constructing the free space of a polyhedron R that rotates around a fixed axis and translates freely relative to a polyhedron O (Secs. 3 and 8). For example, R models a drone helicopter and O models a warehouse. The configuration of R is specified by its position and its rotation angle. The configuration space is the set of configurations. The free space is the subset of the configuration space where the interiors of R and O are disjoint. Robust and efficient free space construction software would advance motion planning, part layout, assembly planning, and mechanical design.

The structure of the free space is determined by the configurations where R has four contacts with O. A contact is determined by a vertex of O and a facet of R, a facet of O and a vertex of R, or an edge of O and an edge of R. The angle (under a rational parameterization) of a configuration with four contacts is a zero of a univariate polynomial of degree 6, which we call an *angle polynomial*, whose coefficients are polynomials in the 48 coordinates of the 16 vertices of the 4 contacts. Every predicate in free space construction is expressible as the sign of an angle polynomial f evaluated at a zero t of an angle polynomial g (Sec. 4). A predicate is degenerate when t is a zero of a common factor h of f and g. It is an identity when h corresponds to a common factor of f and g considered as multivariate polynomials in t and in the vertex coordinates.

Neither prior identity detection approach is practical. Detecting an identity as a zero of the greatest common divisor of f and g is slow (Sec. 9). Devising identity detection logic for every predicate is infeasible because there are over 450,000,000 predicates and 13,000 identities (Sec. 7.4). We present an efficient identity detection algorithm (Sec. 6) based on a one-time analysis of the angle polynomials (Sec. 7).

We enumerate the angle polynomials using canonical variables for the vertex coordinates. Working in the monomial basis is impractical because many of the angle polynomials have over 100,000 terms. Instead, we represent angle polynomials with a data structure, called an *a-poly*, that is a list of sets of vertices (Sec. 5). The enumeration yields 1,000,000 a-polys, which we reduce to the 30,000 representatives of an equivalence relation that respects factorization. We construct a table of factors for the equivalence class representatives in one CPU-day on a workstation.

We factor an angle polynomial by looking up the factors of its representative in the table and substituting its vertex coordinates for the canonical variables. We use the factoring algorithm to associate each zero of an angle polynomial g with an irreducible factor h. Before evaluating a predicate at t, we factor its angle polynomial f. The predicate is an identity if h is one of the factors. Our algorithm is 3500 times faster than computing greatest common divisors. It reduces the share of degeneracy detection in our free space computations from 90% to 0.5% of the running time (Sec. 9). Sec. 10 provides guidelines for applying table-based identity detection to other domains.

2 Prior work

Identity detection is the computational bottleneck in prior work by Hachenberger [2] on computing the Minkowski sum of two polyhedra. He partitions the polyhedra into convex components and forms the union of the Minkowski sums of the components. Neighboring components share common, collinear, or coplanar features, resulting in many identities in the union operations. Detecting the identities via the numerical approach (using CGAL) dominates the running time.

Mayer et al [5] partially compute the free space of a convex polyhedron that rotates around a fixed axis and translates freely relative to a convex obstacle. They report no identity detection problems. Identities can be detected using one rule: all polynomials generated from a facet of one polyhedron and an edge of the other are the same up to sign and hence their zeros are identical. These polynomials correspond to our type I predicates for general polyhedra (Sec. 8).

We address identities in four prior works. We [4] compute polyhedral Minkowski sums orders of magnitude faster than Hachenberger by using a convolution algorithm, which has fewer identities, and by detecting identities with special case logic. We [6] compute free

spaces of planar parts bounded by circular arcs and line segments. The number of identities is small, but the proof of completeness is lengthy. We [7] compute free spaces of polyhedra where R translates in the xy plane and rotates around the z axis. The identity detection logic is retrospectively confirmed using our new approach. There are 816 equivalence classes with 290 in the basis versus 30,564 and 15,306 for the 4D configuration space (Sec. 7).

Finally, we [8] find placements for three polyhedra that translate in a box. The algorithm performs a sequence of ten Minkowski sums and Boolean operations, resulting in many identities. One implementation handles the identities as special cases. A second implementation prevents identities with a polyhedron approximation algorithm that rounds and perturbs the output of each step. The former is twice as fast as the latter and is exact, but took months to develop and lacks a completeness proof.

3 Free space

This section begins our treatment of free space construction. The polyhedra R and O have triangular facets. Without loss of generality, we use the z axis as the axis of rotation. We represent the rotation angle using a rational parameterization of the unit circle. A configuration c of R is a rotation parameter t and a translation vector d, denoted $c = (t, d)$. It maps a point p to the point $c(p) = d + \Theta(t)p$ with

$$\Theta(t)p = \left(\frac{(1-t^2)p_x - 2tp_y}{1+t^2}, \frac{2tp_x + (1-t^2)p_y}{1+t^2}, p_z \right). \tag{1}$$

A point set P maps to $c(P) = \{c(p) \mid p \in P\}$. The free space is $\{c \mid O \cap c(R) = \emptyset\}$.

The boundary of the free space consists of *contact configurations* c at which the boundaries of $c(R)$ and O intersect but not the interiors. The generic contacts are a vertex r_k of R on a facet $o_h o_i o_j$ of O, a vertex o_h of O on a facet $r_i r_j r_k$ of R, and an edge $o_h o_i$ of O tangent to an edge $r_j r_k$ of R. The boundary has faces of dimension $k = 0$ through $k = 3$. A face of dimension k consists of configurations where $4 - k$ contacts occur.

A necessary condition for contact is that the four vertices of the two features are coplanar, so their tetrahedron has zero volume. We substitute the vertices—applying c to those from R—into the volume formula $(q - p) \times (u - p) \cdot (v - p)/6$ to obtain a *contact expression*. We substitute Eq. (1), multiply by 6, and clear the denominator of $1 + t^2$ to obtain a *contact polynomial*, denoted $o_h o_i o_j - r_k$, $o_h - r_i r_j r_k$, or $o_h o_i - r_j r_k$ (Table 1).

Table 1 Contact polynomials.

denotation	contact expression	with
$o_h o_i o_j - r_k$	$d \cdot u + u \cdot \Theta(t) r_k - u \cdot o_j$	$u = (o_i - o_j) \times (o_h - o_j)$
$o_h - r_i r_j r_k$	$d \cdot \Theta(t) u + u \cdot r_k - o_h \cdot \Theta(t) u$	$u = (r_i - r_k) \times (r_j - r_k)$
$o_h o_i - r_j r_k$	$d \cdot (u \times \Theta(t) v) + u \cdot \Theta(t) w + (u \times o_i) \cdot \Theta(t) v$	$u = o_h - o_i$, $v = r_j - r_k$, $w = r_j \times r_k$

Computing the common zeros of four contact polynomials is a core task in free space construction. The polynomials have the form $k_{i1} d_x + k_{i2} d_y + k_{i3} d_z + k_{i4} = 0$ where the k_{ij} are polynomials in t. They have a common zero at $t = t_0$ if the determinant $|k_{ij}|$ is zero and the matrix $[k_{ij}]$ has a nonzero 3-by-3 left minor. We construct the faces of the free space boundary with a sweep algorithm whose events are zeros of these determinants (Sec. 8). Moreover, the vertices are common zeros of contact polynomials, as we explain next.

Figure 2 depicts a zero of the contact polynomials $p_1 = o_0 o_1 o_2 - r_1$, $p_2 = o_1 o_2 o_3 - r_1$, $p_3 = o_0 - r_0 r_1 r_2$, and $p_4 = o_0 - r_0 r_1 r_3$. For this to be a vertex, $c(r_0)c(r_2)$ cannot pierce $o_0 o_1 o_2$ or else the interiors of O and $c(R)$ would intersect. The edge/facet piercing test uses

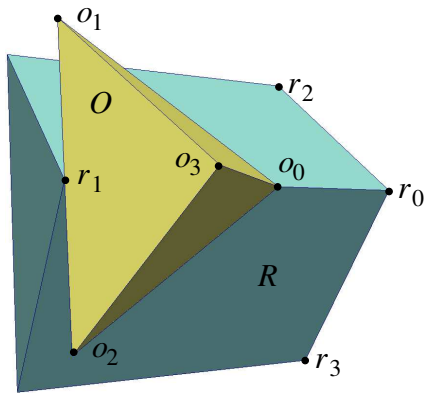


Figure 2 Identity in free space vertex predicate: r_0 is in the plane of $o_0o_1o_2$.

the signs of five contact polynomials, including $o_0o_1o_2 - r_0$. The p_1 and p_2 contacts imply that $c(r_1)$ is on the line of o_1o_2. The p_3 and p_4 contacts imply that o_0 is on the line of $c(r_0)c(r_1)$. Since the line of $c(r_0)c(r_1)$ shares two points with the plane of $o_0o_1o_2$, they are coplanar, $c(r_0)$ is in this plane, and so $o_0o_1o_2 - r_0$ is identically zero. This identity resembles the signed area identities (Sec. 1) in that a polynomial is evaluated on arguments that are derived from the input. However, we cannot apply polynomial identity detection [9] because the arguments are not rational functions of the input but rather the zeros of polynomials whose coefficients are rational functions of the input.

4 Predicates

An *angle polynomial* is a polynomial in t that is used in free space construction. We show that every predicate is expressible as the sign of an angle polynomial f evaluated at a zero of an angle polynomial g. The only exception is the sign of a contact polynomial p evaluated at a common zero c of contact polynomials $\{p_1, p_2, p_3, p_4\}$. We express this form like the other predicates by constructing a polynomial f such that $p(c) = 0$ iff $f(t_0) = 0$ as follows.

Let f be the determinant of $\{p_i, p_j, p_k, p\}$, with $\{p_i, p_j, p_k\}$, $1 \le i < j < k \le 4$, having a non-zero left 3-by-3 minor at $t = t_0$. If $f(t_0) = 0$, $\{p_i, p_j, p_k, p\}$ are linearly dependent in d at $t = t_0$, so p is a linear combination of $\{p_i, p_j, p_k\}$ and $p(c) = 0$. If $p(c) = 0$, $\{p_i, p_j, p_k, p\}$ must be linearly dependent at $t = t_0$ because $p_i(c) = p_j(c) = p_k(c) = 0$ and hence $f(t_0) = 0$.

A degenerate predicate has a polynomial f such that $f(t_0) = 0$ for $t = t_0$ a zero of g. In other words, f and g have a common factor h and $h(t_0) = 0$. This degeneracy is an identity if h results from a common factor of f and g considered as multivariate polynomials in t and in the canonical vertex coordinates. We make such common factor detection fast by enumerating the canonical polynomials, factoring them, and storing the results in a table.

5 Angle polynomials

We represent an angle polynomial with an *a-poly:* a list of elements of the form $L_O - L_R$ with L_O and L_R lists of vertices of O and of R in increasing index order. Elements are in order of increasing $|L_O| + |L_R|$, then increasing $|L_O|$, then increasing vertex index (lexicographically). There are three kinds of a-polys.

The first kind represents the angle polynomials at whose zeros four contacts can occur. It has three types of elements. A 1-contact denotes a contact polynomial. A 2-contact $o_h - r_ir_j$ denotes 1-contacts $o_h - r_ir_jr_k$ and $o_h - r_ir_jr_l$ that jointly constrain o_h to the line

$r_i r_j$; likewise $o_h o_i - r_j$. A 3-contact $o_h - r_i$ denotes three polynomials whose zeros are the configurations where the two vertices coincide. A list of elements that together denote four polynomials comprises an a-poly.

The second kind represents an angle polynomial that is zero when an edge of one polyhedron and two edges of the other polyhedron are parallel to a common plane. It has three elements of types $o_i o_j -$ and $-r_i r_j$, for example $(o_1 o_2 -, -r_1 r_2, -r_4 r_5)$. However, if the two edges share a vertex, we contract $(o_h o_i -, -r_j r_k, -r_j r_l)$ to $(o_h o_i - r_j r_k r_l)$, corresponding to an edge parallel to a facet. Likewise, $(o_h o_i o_j - r_k r_l)$. The third kind corresponds to the 3-by-3 left minor (Sec. 3): the d-coefficients of three contact polynomials. The d-coefficients of $o_i o_j o_k - r_l$ and $o_i - r_i r_j r_k$ do not depend on r_l and o_i, hence the elements are of type $o_i o_j o_k -$, $-r_i r_j r_k$, or $o_h o_i - r_j r_k$.

The derivation of the angle polynomials from their a-polys is as follows.

Four 1-contacts. The contact expressions (Table 1) have the form $d \cdot n + k$. The vectors n have the form a, $\Theta(t)a$, or $a \times \Theta(t)b$ and the summands of the scalar k have the form $a \cdot b$ or $a \cdot \Theta(t)b$ with a and b constant vectors. The angle polynomial is the numerator of

$$\begin{vmatrix} n_{1x} & n_{1y} & n_{1z} & k_1 \\ n_{2x} & n_{2y} & n_{2z} & k_2 \\ n_{3x} & n_{3y} & n_{3z} & k_3 \\ n_{4x} & n_{4y} & n_{4z} & k_4 \end{vmatrix}.$$

Expanding in terms of minors using the last column, k_1 has minor $n_2 \cdot (n_3 \times n_4)$. Using the formula, $u \times (v \times w) = (u \cdot w)v - (u \cdot v)w$, each minor reduces to a sum of terms of the form $a \cdot b$, $a \cdot \Theta(t)b$, or $(a \cdot \Theta(t)b)(c \cdot \Theta(t)d)$. Applying equation (1) and clearing the denominator reduces k_1, k_2, and k_3 to quadratics in t and reduces the minors to quartics in t, so the angle polynomial has degree 6.

One 2-contact and two 1-contacts. For a 2-contact $o_i o_j - r_k$, $c(r_k) = d + \Theta(t)r_k$ is on the line of $o_i o_j$, so d is on the line through $o_i - \Theta(t)r_k$ and $o_j - \Theta(t)r_k$. We intersect this line with the planes of the other two contact polynomials. We express the line as $\lambda u + v$ with $u = o_j - o_i$ and $v = o_j - \Theta(t)r_k$, compute the values $\lambda_i = -(n_i \cdot v + k_i)/(n_i \cdot u)$ where the line intersects the two planes $n_i \cdot p + k_i$, set $\lambda_1 = \lambda_2$, and cross multiply to obtain a quartic angle polynomial. Similarly 2-contact $o_i - r_j r_k$ corresponds to a line with $u = \Theta(t)(r_j - r_k)$ and $v = o_i - \Theta(t)r_j$.

Two 2-contacts. The expression is the signed volume of the four points that define the lines of the 2-contacts, which yields a quartic angle polynomial. Figure 2 illustrates this situation.

One 3-contact and one 1-contact. For a 3-contact $o_i - r_j$, $o_i = d + \Theta(t)r_j$. We substitute $d = o_i - \Theta(t)r_j$ into the 1-contact polynomial to obtain a quadratic angle polynomial.

Other kinds. The second, $(-r_{i_1} r_{j_1}, -r_{i_2} r_{j_2}, o_{i_3} o_{j_3} -)$ and $(-r_{i_1} r_{j_1}, o_{i_2} o_{j_2} -, o_{i_3} o_{j_3} -)$, has expressions that yield quadratic angle polynomials $\Theta(t)((r_{j_1} - r_{i_1}) \times (r_{j_2} - r_{i_2})) \cdot (o_{j_3} - o_{i_3})$ and $\Theta(t)(r_{j_1} - r_{i_1}) \cdot ((o_{j_2} - o_{i_2}) \times (o_{j_3} - o_{i_3}))$. The second has expression $(n_1 \times n_2) \cdot n_3$, where $o_i o_j o_k -$ has normal $n = (o_j - o_i) \times (o_k - o_i)$, $-r_i r_j r_k$ has normal $n = \Theta(t)((r_j - r_i) \times (r_k - r_i))$, and $o_h o_i - r_j r_k$ has normal $n = (o_i - o_h) \times \Theta(t)(r_k - r_j)$. The third has the same polynomials as the quartic minor polynomials above in the four 1-contacts case.

6 Factoring

This section gives an algorithm for factoring angle polynomials represented as a-polys. Two a-polys are *equivalent* if a vertex bijection maps one to the other. The bijection maps a factorization of one to a factorization of the other. The factoring algorithm uses a table that contains the factorization of a representative a-poly from each equivalence class. Sec. 7 explains how we constructed this table. It is available in the web directory `http://www.cs.miami.edu/~vjm/robust/identity`.

Identity detection requires that an irreducible polynomial be denoted by a unique a-poly, so one can detect if different a-polys have a common factor. One problem is that nonequivalent a-polys can denote the same polynomial. We solve this problem by selecting the factors in the table from a minimal subset of the equivalence classes that we call *basis* classes. A second problem is that equivalent a-polys can denote the same polynomial. This problem is so rare that we can record all the basis a-polys that generate a factor, called its *factor set,* and select a unique one during factoring. The factoring algorithm maps an input a-poly to the representative of its equivalence class, obtains the factor sets of the representative from the table, and applies the inverse map to obtain sets of a-polys in the variables of the input a-poly. To achieve uniqueness, it selects the lexicographical minimum from each set, using a vertex order that we indicate by the o and r indices.

6.1 Mapping an a-poly to its representative

The mapping algorithm generates the permutations of the input a-poly such that the elements remain increasing in $|L_O| + |L_R|$ then in $|L_O|$ (but disregarding the lexicographical order of vertex indices). For each permuted contact list, it assigns each vertex an indicator: a bit string in which a 1 in position k indicates that the vertex appears in the kth element. It labels a permutation with its O vertex indicators in decreasing order followed by its R vertex indicators in decreasing order. It selects the permutation with the largest label, replaces the ith O vertex in indicator order by the canonical vertex o_i, and likewise for the R vertices and r_i. If two vertices have the same indicator, both orders yield the same output because the indices of an a-poly are placed in increasing order.

For example, in a-poly $(o_{27} - r_{22}r_{66}r_{86}, o_{43} - r_{22}r_{66}r_{86}, o_{27}o_{51} - r_{15}r_{86}, o_{27}o_{43} - r_{75}r_{86})$, o_{27} has indicator 1011, and the indicator list is 1011,0101,0010;1111,1100,1100,0010,0001. The permutations swap the first two and/or last two elements. Swapping the last two yields the maximal indicator list 1011,0110,0001;1111,1100,1100,0010,0001 and the representative $(o_0 - r_0r_1r_2, o_1 - r_0r_1r_2, o_0o_1 - r_0r_3, o_0o_2 - r_0r_4)$. In the table, this representative has factor sets $\{(o_0o_1 - r_0r_1r_2)\}$ and $\{(-o_0o_1o_2, o_0o_1 - r_0r_3, o_0o_2 - r_0r_4)\}$. The inverse vertex mapping yields the factors $(o_{27}o_{43} - r_{22}r_{66}r_{86})$ and $(-r_{22}r_{66}r_{86}, o_{27}o_{43} - r_{75}r_{86}, o_{27}o_{51} - r_{15}r_{86})$.

6.2 Uniqueness

Fig. 3 illustrates how a-polys can have the same polynomial. The 1-contacts $o_0o_1 - r_0r_1$, $o_0o_1 - r_2r_3$, and $o_0o_1o_2 - r_4$ determine the angle parameter t because they are invariant under translation parallel to o_0o_1. Translating R parallel to o_0o_1 until one element becomes a 2-contact yields an a-poly C_i that is zero at the same t values. If r_0r_1 hits o_0, $C_1 = (o_0 - r_0r_1, o_0o_1 - r_2r_3, o_0o_1o_2 - r_4)$ (Fig. 3b) and if r_2r_3 hits o_1, $C_2 = (o_1 - r_2r_3, o_0o_1 - r_0r_1, o_0o_1o_2 - r_4)$. These a-polys are equivalent under the map from C_1 to C_2: $o_0 \to o_1$, $o_1 \to o_0$, $o_2 \to o_2$, $r_0 \to r_2$, $r_1 \to r_3$, $r_2 \to r_0$, $r_3 \to r_1$, $r_4 \to r_4$. There are also C_3 and C_4 where r_0r_1 hits o_1 or r_2r_3 hits o_0 and $C_5 = (o_0o_2 - r_4, o_0o_1 - r_0r_1, o_0o_1 - r_2r_3)$ and

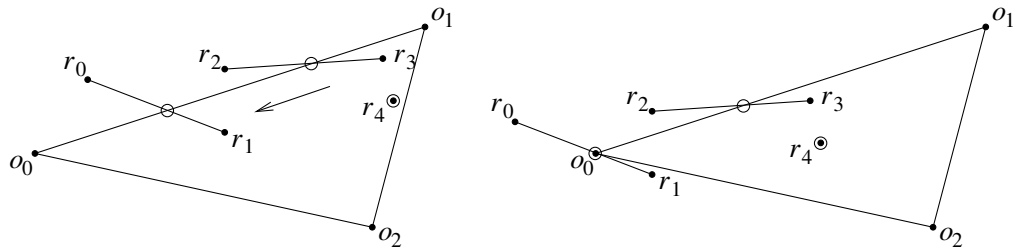


Figure 3 Equivalent a-polys: (a) translation parallel to o_0o_1 preserves circled contacts $o_0o_1 - r_0r_1$, $o_0o_1 - r_2r_3$, and $o_0o_1o_2 - r_4$; (b) a-poly $(o_0 - r_0r_1, o_0o_1 - r_2r_3, o_0o_1o_2 - r_4)$.

$C_6 = (o_1o_2 - r_4, o_0o_1 - r_0r_1, o_0o_1 - r_2r_3)$ where r_4 hits o_0o_2 or o_1o_2. C_5 and C_6 are not equivalent to C_1, C_2, C_3, and C_4 because their first element has two O vertices, not one.

The a-poly $(o_1 - r_0r_1r_2, o_0o_1 - r_3r_4, o_0o_1 - r_5r_6, o_oo_1o_2 - r_7)$ maps to the representative $(o_0 - r_0r_1r_2, o_0o_1 - r_3r_4, o_0o_1 - r_5r_6, o_oo_1o_2 - r_7)$ with factor sets $\{(o_0o_1 - r_0r_1r_2)\}$ and

$$\{(o_0 - r_3r_4, o_0o_1 - r_5r_6, o_0o_1o_2 - r_7), (o_0 - r_5r_6, o_0o_1 - r_3r_4, o_0o_1o_2 - r_7),$$
$$(o_1 - r_3r_4, o_0o_1 - r_5r_6, o_0o_1o_2 - r_7), (o_1 - r_5r_6, o_0o_1 - r_3r_4, o_0o_1o_2 - r_7)\}$$

because the second factor is equivalent to $C_1, \ldots, C_4$. Its factor set does not contain a-polys equivalent to C_5 and C_6 because their class is not in the basis. The inverse map (in this case swapping o_0 and o_1) results in factor sets $\{(o_0o_1 - r_0r_1r_2)\}$ and

$$\{(o_1 - r_3r_4, o_0o_1 - r_5r_6, o_0o_1o_2 - r_7), (o_1 - r_5r_6, o_0o_1 - r_3r_4, o_0o_1o_2 - r_7),$$
$$(o_0 - r_3r_4, o_0o_1 - r_5r_6, o_0o_1o_2 - r_7), (o_0 - r_5r_6, o_0o_1 - r_3r_4, o_0o_1o_2 - r_7)\}.$$

The algorithm selects the (minimal) third element $(o_0 - r_3r_4, o_0o_1 - r_5r_6, o_0o_1o_2 - r_7)$ of the second set as the second factor.

7 Constructing the table of factors

This section explains how we enumerate the a-poly equivalence classes (Sec. 7.1), factor the class representatives and select a basis (Sec. 7.2), and construct the table of factors (Sec. 7.3). The factoring algorithm is probabilistic and depends on the completeness assumption that the factors of a-polys are a-polys. If this assumption were false, the algorithm would have failed. We verify the table to a high degree of certainty using standard techniques for testing polynomial identities (Sec. 7.4).

7.1 Equivalence classes

We enumerate the a-poly classes as follows. Let a_i, b_i, c_i, d_i, e_i, and f_i denote vertices and define $s_i = \{a_i - d_i\}$, $t_i = \{a_i - d_ie_i, a_ib_i - d_i\}$, $u_i = \{a_i - d_ie_if_i, a_ib_i - d_ie_i, a_ib_ic_i - d_i\}$, $v_i = \{a_ib_i - c_id_ie_i, a_ib_ic_i - d_ie_i\}$, $w_i = \{-a_ib_i, a_ib_i-\}$, $x_i = \{-a_ib_ic_i, a_ib_i - c_id_i, a_ib_ic_i-\}$. We generate the a-polys that are lists of k-contacts with the sets $s_1 \times u_2$ (a 3-contact and a 1-contact), $t_1 \times t_2$ (two 2-contacts), $t_1 \times u_2 \times u_3$ (a 2-contact and two 1-contacts), and $u_1 \times u_2 \times u_3 \times u_4$ (four 1-contacts). We generate the other kinds of a-polys with the sets v_1 (edge parallel to facet), $w_1 \times w_2 \times w_3$ (edges parallel to plane), and $x_1 \times x_2 \times x_3$ (3-by-3 left minor). For each element of each set, we assign O vertices to the a_i, b_i, c_i in every possible

manner. Starting from o_0, we assign increasing indices to the vertices of an edge or a facet. We assign R vertices to d_i, e_i, f_i likewise. The highest index is 11.

For example, $s_1 \times u_2 = \{(a_1 - d_1, a_2 - d_2e_2f_2), (a_1 - d_1, a_2b_2 - d_2e_2), (a_1 - d_1, a_2b_2c_2 - d_2)\}$. We must set $a_1 = o_0$. We can set $a_2 = o_0$ or $a_2 = o_1$ because a_2 is in a different feature, and then assign increasing indices to b_2 and c_2. Similarly for d_1, d_2, e_2, and f_2. The results are
$\{(o_0 - r_0, o_0 - r_0r_1r_2), (o_0 - r_0, o_1 - r_0r_1r_2), (o_0 - r_0, o_0 - r_1r_2r_3), (o_0 - r_0, o_1 - r_1r_2r_3),$
$(o_0 - r_0, o_0o_1 - r_0r_1), (o_0 - r_0, o_1o_2 - r_0r_1), (o_0 - r_0, o_0o_1 - r_1r_2), (o_0 - r_0, o_1o_2 - r_1r_2),$
$(o_0 - r_0, o_0o_1o_2 - r_0), (o_0 - r_0, o_1o_2o_3 - r_0), (o_0 - r_0, o_0o_1o_2 - r_1), (o_0 - r_0, o_1o_2o_3 - r_1)\}$.

The enumeration yields about one million a-polys. Generating their representatives (Sec. 6) and removing duplicates yields 30,564 equivalence class representatives.

7.2 Basis classes

We factor the representatives probabilistically. We replace the canonical coordinates of $o_0, \ldots, o_{11}$ and $r_0, \ldots, r_{11}$ with random integers, construct the resulting univariate integer polynomials, and factor them with Mathematica. An irreducible univariate implies that the canonical polynomial is irreducible; the converse is true with high probability.

An a-poly depends on a vertex if its univariate changes when the coordinates of the vertex are assigned different random integers. For example, $(o_0o_1 - r_0, o_2 - r_1r_2r_3, o_2 - r_1r_2r_4)$ does not depend on r_3 and r_4 because $o_2 - r_1r_2r_3$ and $o_2 - r_1r_2r_4$ can be replaced by $o_2 - r_1r_2$: o_2 in contact with $r_1r_2r_3$ and $r_1r_2r_4$ is equivalent to o_2 in contact with r_1r_2. An a-poly is *complete* if it depends on all of its vertices. An angle polynomial is *contiguous* if it depends on $o_0, o_1, \ldots, o_l$ and $r_0, r_1, \ldots, r_m$ for some l and m. A complete representative is also contiguous because we assign vertices to the indicators contiguously (Sec. 6).

We select a basis of complete, contiguous, and irreducible a-polys, represented by the representatives of their classes. (We prove that such a basis exists by verifying the table of factors (Sec. 7.4).) We construct a map I from the univariate of each basis a-poly to the set of basis a-polys that generate it. In the Sec. 6.2 example, C_1 is in the basis and generates a univariate p. The equivalent a-polys C_2, C_3, and C_4 also generate p, so $I(p) = \{C_1, C_2, C_3, C_4\}$. Although C_5 and C_6 also generate p, they are not in $I(p)$ because they belong to another (necessarily non-basis) equivalence class.

The algorithm visits each representative ρ. If ρ is complete and its univariate p is irreducible but $I(p) = \emptyset$, the algorithm adds ρ to the basis representatives, permutes its vertices in every way, calculates the univariate u for each resulting a-poly a, and adds a to the set $I(u)$.

The condition $I(p) = \emptyset$ prevents adding an a-poly to the basis whose univariate is already generated by a member of a basis class. For example, $\rho_1 = (-r_0r_1r_2, -r_0r_3r_4, o_0o_1 - r_0r_5)$ is assigned to the basis. Later, $\rho_2 = (o_0 - r_0r_1, o_0 - r_2r_3r_4, o_0o_1 - r_2r_5)$ is complete and has an irreducible univariate that is generated by $(-r_0r_1r_2, -r_2r_3r_4, o_0o_1 - r_2r_5)$, which is a permutation of ρ_1. Since every permutation of ρ_1 is in I, ρ_2 is not assigned to the basis.

7.3 Factor table

The factor table provides a list of factor sets for each representative. For a basis representative ρ with univariate p, the list is $\langle I(p) \rangle$. If ρ is not in the basis, we process each factor f of p as follows.

(1) Determine which vertices f depends on. Randomly change a vertex of ρ, generate the new univariate, and factor it. If f is not a factor, it depends on the vertex.

(2) Rename the vertices of ρ to obtain a ρ' for which the factor f' that corresponds to f is contiguous. Let f depend on $o_{i_0}, o_{i_1}, \ldots, o_{i_{m'}}$ and $r_{j_0}, r_{j_1}, \ldots, r_{j_{n'}}$ but not on $o_{i_{m'+1}}, o_{i_{m'+2}}, \ldots, o_{i_m}$ and $r_{j_{n'+1}}, r_{j_{n'+2}}, \ldots, r_{j_n}$. Substitute $o_{i_k} \to o_k$ for $k = 1, \ldots, m$ and $r_{i_k} \to r_k$ for $k = 1, \ldots, n$.

(3) Find the factor f' of ρ' that depends on $o_0, \ldots, o_{m'}$ and $r_0, \ldots, r_{n'}$, and has the same degree as f. (This factor is unique; otherwise, we would consider every match.)

(4) Look up $I(f')$ and invert the vertex substitution to obtain a factor set.

For example, the univariate of $\rho = (o_0o_1 - r_0r_1, o_2o_3o_4 - r_2, o_2o_3o_4 - r_3, o_5o_6o_7 - r_4)$ has a factor f that depends on all its variables and a quadratic factor g that depends on o_2, o_3, o_4, r_2, r_3. The factor set of f is $I(f) = \{(o_2o_3o_4-, o_5o_6o_7-, o_0o_1 - r_0r_1)\}$. To obtain the factor set of g, substitute $o_2, o_3, o_4, o_0, o_1, r_2, r_3, r_0, r_1 \to o_0, o_1, o_2, o_3, o_4, r_0, r_1, r_2, r_3$. The quadratic factor g' of $\rho' = (o_3o_4 - r_2r_3, o_0o_1o_2 - r_0, o_0o_1o_2 - r_1, o_5o_6o_7 - r_4)$ depends on $o_0o_1o_2$ and r_0r_1, and $I(g') = \{(o_0o_1o_2 - r_0r_1)\}$. Inverting the vertex substitution yields the second factor set of ρ: $\{(o_2o_3o_4 - r_2r_3)\}$.

To save space, we do not add entries to I corresponding to permutations of basis representatives with degree-6 univariates because they cannot be proper factors. To test if ρ with irreducible degree-6 univariate p is in the basis, we generate the permutations of ρ and their univariates. If none has an entry in I, ρ is in the basis, and we add an entry for p to I. If the univariate p' of a permutation has an entry in I, the sole factor set of ρ is the result of applying the inverse of the permutation to $I(p')$.

7.4 Analysis

Out of 30,564 representatives, 15,306 are basis, 991 are constant, 3840 are irreducible but non-basis, 8263 have two factors (including 260 squares), and 2164 have three factors (including 6 cubes). Since a predicate is an a-poly evaluated on a zero of a basis polynomial, we listed $450,000,000 \approx 30,564 \cdot 15,306$ predicates in the introduction. Likewise, we stated the number of identities as $13,000 \approx 1 \cdot 3840 + 2 \cdot 8263 + 3 \cdot 2164$ ways of evaluating a non-basis polynomial on the zero of a factor. Of the irreducible representative polynomials, 363 have two basis a-polys, 50 have three, 194 have four, and 194 have six.

Each factorization $f_1f_2 \cdots f_m | f$ is equivalent to a polynomial identity $f - af_1f_2 \cdots f_m = 0$ for some constant a. Instead of analyzing the probability of failure of the algorithm, we verify the identities probabilistically using Schwartz's lemma [9]. We use random 20-bit values modulo a prime p. The first substitution determines a and the rest verify the identity. Verifying all 15,258 factorizations once takes 2 seconds and 10 minutes of verification reduces the probability of an error to below 10^{-1000}. This also constitutes a probabilistic proof of the completeness assumption.

The running time for factor table construction was one CPU day. All but one CPU hour was spent generating the permutations of the degree-six irreducible polynomials to test if they are in the basis. The worst case is four contacts between O vertices and R facets or vice versa, which have about 70 billion permutations. The tests all succeeded, so perhaps we could have avoided this cost by proving a theorem.

8 Contact set subdivision

We summarize the portion of the freespace construction used to measure the effect of identity detection. The details appear in the full version of the paper. The *contact facet* of a vertex o_h of O and a facet $r_ir_jr_k$ of R is the set of configurations c such that o_h is tangent to $c(r_ir_jr_k)$, disregarding overlap of O and $c(R)$ elsewhere. We denote this

contact facet as $o_h - r_i r_j r_k$ because of its close relation to that a-poly. For fixed t, d lies on the triangle $o_h - \Theta(t) r_i, o_h - \Theta(t) r_j, o_h - \Theta(t) r_k$. Likewise $o_h o_i o_j - r_k$ (triangle) and $o_h o_i - r_j r_k$ (parallelogram). Contact facet $o_h - r_i r_j r_k$ has *contact edge* $o_h - r_i r_j$, $d \in o_h - \Theta(t) r_i, o_h - \Theta(t) r_j$, *contact vertex* $o_h - r_i$, $d = o_h - \Theta(t) r_i$, and so forth. We use F, E, and V as shorthand for contact face, edge, and vertex. An EF is the intersection of a contact edge and contact facet, and likewise an FF and an FFF. The algorithm creates a-polys from the denotations. For example, (E,F,F) would be the a-poly with one 2-contact and two 1-contacts resulting from listing the denotations of a contact edge and two contact facets.

We trace the evolution of the arrangment of EFs, FFs, and FFFs on a single contact facet as t ranges over the interval it exists. The events have four types (below). After each event, we update the list of EFs on its contact edges, and the list of FFFs on each FF. Ordering two EFs on an E can be expressed as a-poly (E,F,F). Checking if an EF lies on an E uses two (V,F) a-polys. Ordering two FFFs on an FF has a-poly (F,F,F,F).

Type I: An F and its induced EFs, FFs, and FFFs appear or disappear at the zero of a parallel edge/facet a-poly. Type II: an FF and its induced EFs and FFFs appear or disappear at a (V,F) or (E,E) a-poly zero. Type III: an FFF appears or disappears on an FF and two EFs swap on an E at an (E,F,F) zero. Type IV: FFFs swap on FFs at an (F,F,F,F) zero.

Before evaluating a predicate at an event angle, we check for identity. An identity results from evaluating the predicate at a parameter t that is a zero of a $k \geq 1$ repeated irreducible factor of the a-poly of the predicate. We replace the sign of the predicate with the sign of its kth derivative, evaluated using automatic differentiation. The derivative gives the sign of the predicate value immediately after the event as required.

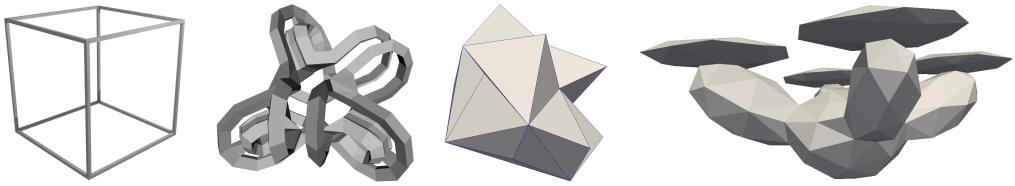
9 Results

We measure the running time of predicate evaluation using the factoring algorithm (Sec. 6) for identity detection and compare it to using the greatest common divisor (GCD) for degeneracy detection.

We compute the GCD with Euclid's algorithm. The main step is polynomial division. We compute the degree of a remainder by finding its nonzero coefficient of highest degree. If the sign of a coefficient is ambiguous in double precision interval arithmetic, we evaluate modulo several 32-bit primes and invoke the Chinese Remainder Theorem (CRT). The a-polys have degree at most 9 in the input coordinates (Sec. 5). The leading coefficient is at most degree 9 in the polynomial coefficients. Hence, CRT requires $\lceil 9 \cdot 9 \cdot 53/32 \rceil = 135$ modulo evaluations. If they are all zero, the coefficient is zero. This analysis assumes that all inputs have the same exponent field and can be replaced by their 53-bit integer mantissas.

In our first set of tests, we selected 100,000 representatives at random and instantiated them on a pool of 12 O vertices and 12 R vertices with random coordinates in $(-1, 1)$. We factored the univariates of these a-polys, isolated the zeros of the factors, and stored them in a red-black tree.

Let t_h be the ith largest real zero of h, a factor of a random a-poly g. When inserting t_h into the tree, it must be compared to prior zeros, such as the jth zero t_e of e, a factor of f. If $e = h$ and $i = j$, then $t_h = t_e$. To measure the running time of the identity detection algorithm (Sec. 6), we add an unnecessary test if $g(t_e)$ is an identity. This ensures identity tests on random polynomials with both possible answers. Adding these 9,660,759 identity tests (221,252 positive) increases the running time from 6.6 seconds to 18.1 seconds, giving an average identity detection time of 1.2 microseconds.



frame: 40/96 knot: 480/992 simple: 18/32 drone: 189/374

Figure 4 Sweep Inputs and number of vertices/facets. Actual frame and knot are large enough for simple and drone to fly through.

Table 2 Sweep algorithm: f total number of contact facets, t_{fac} and t_{GCD} average running time in seconds for sweeping a facet with our identity detection and GCD-based degeneracy detection.

	O	R	f	t_{fac}	t_{GCD}	$t_{\text{GCD}}/t_{\text{fac}}$
1	frame	simple	2196	0.021	0.533	25.33
2	frame	drone	18812	0.148	2.093	14.13
3	knot	simple	23419	0.023	0.651	27.24
4	knot	drone	235789	0.037	0.857	23.26

To test the GCD approach, we drop factoring and the equality test, and replace each polynomial f with its square-free form $f/\text{GCD}(f, f')$. If the comparison of zeros t_f of f and t_g of g is ambiguous in double precision, we run the following degeneracy test on $g(t_f)$. Set $h \leftarrow \text{GCD}(f, g)$ and $e = f/h$. If $e(t_f)$ is unambiguously nonzero, $g(t_f)$ must be zero. If $e(t_f)$ and $g(t_f)$ are both ambiguous, redo these steps with more precision. The additional time was 376 seconds for 81264 degeneracy tests, for an average time of 4627 microseconds. To be sure that $t_f = t_g$ and not some other zero of g, we must check that its comparison with other zeros is unambiguous, so the true cost of the GCD method is even higher.

In the second set of tests, we ran a sweep algorithm (Sec. 8) on the polyhedra shown in Fig. 4. Table 2 shows the average running times for sweeping a facet using factor-based identity detection and GCD degeneracy detection. We sweep either all the facets (tests 1-3) or a large random sample over all angles (test 4).

The first tests indicate that factor-based identity detection is 3500 times faster than GCD-based degeneracy detection. The sweep tests show that the effect of this improvement entails a factor of 14 speedup in sweep time. Since identity detection is sped up by 3500, factor-based identity detection uses less that 0.5% of the overall running time, versus 90% for GCD-based degeneracy detection.

10 Discussion

We have shown that looking up the factors of an a-poly is much faster than polynomial algebra for zero detection. As an additional advantage, factorization provides a unique representation of each algebraic number as the ith zero of an irreducible polynomial.

In future work, we will extend the sweep algorithm to completely construct the subdivision of a contact set. We are missing the surfaces that bound the cells and their nesting order. We will construct a connected component of the free space boundary by visiting the neighboring contact facets, computing their subdivisions, and so on. All the predicates are angle polynomials evaluated at zeros of angle polynomials.

For efficient free space boundary construction, we must eliminate irrelevant contact facets from consideration and must eliminate irrelevant sweep angles for relevant contact facets. One strategy is to construct a polyhedral inner and outer approximation of R as it sweeps through a small angle. For that angle range, the boundary of the rotational free space lies between the boundaries of the translational free spaces of the approximations. We can use our fast polyhedral Minkowski sum software [4] to generate approximations of the rotational free space boundary for a set of angle intervals covering the unit circle. We see no reason that sweeping a relevant facet should have fewer identities than sweeping an irrelevant facet, and so fast identity detection should provide the same speedup as observed in Sec. 9.

We conclude that our identity detection algorithm is useful for the drone–warehouse problem. But does the technique generalize to other domains? We discuss three challenges.

Factor table construction (Sec. 7) depends on the property that the set of polynomials is closed under factorization. What if this is not true for some alternate class of polynomials? We would realize that something was missing when factor table construction failed due to unmatched univariates. We would then analyze the failure to uncover the missing polynomials. (This is how we discovered the three-edge parallel-feature a-polys.) The new polynomials might also have had unmatched factors. However, the process of adding missing factor polynomials must converge because factoring reduces degree.

Representative generation (Sec. 6) depends strongly on the a-poly representation. However, the approach should generalize. A predicate has symmetries on its inputs that leave it alone or flip its sign. Two invocations of a predicate function (with repeated inputs) are isomorphic if one can get from one to the other by applying those symmetries and reindexing inputs. A representative is the isomorphism class member that is lexicographically minimal.

Generalizing the matching of factors to prior classes for table generation would greatly increase the running time because it uses enumeration of permutations. In free space construction for R with d degrees of freedom, the computational complexity is $(3d!)d^2$, which would balloon to 80,000 years for $d = 6$ (unconstrained rotation and translation). We could use the "Birthday paradox" to match two polynomials by generating the square root of the number of permutations for each one and finding a collision, reducing the running time by a factor of $\sqrt{3d!}$ to less than a day. Moreover, these enumerations can be tested in parallel.

Greater complexity might also increase the time required to construct a representative (Sec. 6) for identity detection, but this cost can be reduced by using a subset of the symmetry group and increasing the size of the lookup table. For example, if we did not reorder the elements, the table size would be 134,082, and if we did not reorder the left and right columns either, it would be 972,806. These tables are easy to generate from the minimal table, and both are reasonable for current RAM sizes.

References

1 CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

2 Peter Hachenberger. Exact Minkowski sums of polyhedra and exact and efficient decomposition of polyhedra into convex pieces. *Algorithmica*, 55:329–345, 2009.

3 Dan Halperin. Controlled perturbation for certified geometric computing with fixed-precision arithmetic. In *ICMS*, pages 92–95, 2010.

4 Min-Ho Kyung, Elisha Sacks, and Victor Milenkovic. Robust polyhedral Minkowski sums with GPU implementation. *Computer-Aided Design*, 67–68:48–57, 2015.

5 Naama Mayer, Efi Fogel, and Dan Halperin. Fast and robust retrieval of Minkowski sums of rotating convex polyhedra in 3-space. *Computer-Aided Design*, 43(10):1258–1269, 2011.

6 Victor Milenkovic, Elisha Sacks, and Steven Trac. Robust free space computation for curved planar bodies. *IEEE Transactions on Automation Science and Engineering*, 10(4):875–883, 2013.

7 Elisha Sacks, Nabeel Butt, and Victor Milenkovic. Robust free space construction for a polyhedron with planar motion. *Computer-Aided Design*, 90C:18–26, 2017.

8 Elisha Sacks and Victor Milenkovic. Robust cascading of operations on polyhedra. *Computer-Aided Design*, 46:216–220, 2014.

9 J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27(4):701–717, 1980. doi:10.1145/322217.322225.

Approximate Range Queries for Clustering

Eunjin Oh
Max Planck Institute for Informatics
Saarbrücken, Germany
eoh@mpi-inf.mpg.de
https://orcid.org/0000-0003-0798-2580

Hee-Kap Ahn
Pohang University of Science and Technology
Pohang, Korea
heekap@postech.ac.kr
https://orcid.org/0000-0001-7177-1679

Abstract

We study the approximate range searching for three variants of the clustering problem with a set P of n points in d-dimensional Euclidean space and axis-parallel rectangular range queries: the *k-median*, *k-means*, and *k-center range-clustering query problems*. We present data structures and query algorithms that compute $(1+\varepsilon)$-approximations to the optimal clusterings of $P \cap Q$ efficiently for a query consisting of an orthogonal range Q, an integer k, and a value $\varepsilon > 0$.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Approximate clustering, orthogonal range queries

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.62

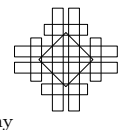
Related Version A full version of this paper is available at `http://arxiv.org/abs/1803.03978`.

Funding This research was supported by NRF grant 2011-0030044 (SRC-GAIA) funded by the government of Korea, and the MSIT (Ministry of Science and ICT), Korea, under the SW Starlab support program (IITP-2017-0-00905) supervised by the IITP (Institute for Information & communications Technology Promotion).

1 Introduction

Range searching asks to preprocess a set of objects and to build a data structure so that all objects intersecting a given query range can be reported quickly. There are classical variants of this problem such as computing the number of objects intersecting a query range, checking whether an object intersects a query range, and finding the closest pair of objects intersecting a query range. It is a widely used technique in computer science with numerous applications in geographic information systems, computer vision, machine learning, and data mining. These range searching problems have been studied extensively in computational geometry over the last decades. For more information on range searching, refer to the surveys [2, 20].

However, there are a large number of objects intersecting a query range in many real-world applications, and thus it takes long time to report all of them. Thus one might want to obtain a property of the set of such objects instead of obtaining all such objects. Queries of this kind are called *range-analysis queries*. More formally, the goal of this problem is to preprocess a set P of objects with respect to a fixed range-analysis function f and to build a data structure so that $f(P \cap Q)$ can be computed efficiently for any query range Q. These query problems have been studied extensively under various range-analysis functions such as




the diameter or width of a point set [8] and the length of the minimum spanning tree of a point set [6]. Note that the classical variants mentioned above are also range-analysis query problems. A clustering cost can also be considered as a range-analysis function.

Clustering is a fundamental research topic in computer science and arises in various applications [19], including pattern recognition and classification, data mining, image analysis, and machine learning. In clustering, the objective is to group a set of data points into clusters so that the points from the same cluster are similar to each other and points from different clusters are dissimilar. Usually, input points are in a high-dimensional space and the similarity is defined using a distance measure. There are a number of variants of the clustering problem in the geometric setting depending on the similarity measure such as the k-median, k-means, and k-center clustering problems.

In this paper, we study the approximate range-analysis query problems for three variants of the clustering with a set P of n points in d-dimensional Euclidean space with $d \geq 2$ and axis-parallel rectangular range queries: the *k-median*, *k-means*, and *k-center range-clustering query problems*. The approximate k-median, k-means and k-center range-clustering query problems are defined as follows: Preprocess P so that given a query range Q, an integer k with $1 \leq k \leq n$ and a value $\varepsilon > 0$ as a query, an $(1+\varepsilon)$-approximation to the k-median, k-means, and k-center clusterings of $P \cap Q$ can be computed efficiently. Our desired query time is polynomial to $\log n$, k and $(1/\varepsilon)$.

Previous work. The k-median and k-means clustering problems have been studied extensively and there are algorithms achieving good approximation factors and polynomial running times to the problem. Har-Peled and Mazumdar [18] presented an $(1+\varepsilon)$-approximation algorithm for the k-means and k-median clustering using coresets for points in d-dimensional Euclidean space. Their algorithm constructs a (k, ε)-coreset with property that for any arbitrary set C of k centers, the clustering cost on the coreset with respect to C is within $(1 \pm \varepsilon)$ times the clustering cost on the original input points with respect to C. Then it computes the clusterings for the coreset using a known weighted clustering algorithm. Later, a number of algorithms for computing smaller coresets for the k-median and k-means clusterings have been presented [9, 13, 17].

The k-center clustering problem has also been studied extensively. It is NP-Hard to approximate the 2-dimensional k-center problem within a factor of less than 2 even under the L_∞-metric [12]. A 2-approximation to the k-center can be computed in $O(kn)$ time for any metric space [12], and in $O(n \log k)$ time for any L_p-metric space [14]. The exact k-center clustering can be computed in $n^{O(k^{1-1/d})}$ time in d-dimensional space under any L_p-metric [4]. This algorithm combined with a technique for grids yields an $(1+\varepsilon)$-approximation algorithm for the k-center problem that takes $O(n \log k + (k/\varepsilon)^{O(k^{1-1/d})})$ time for any L_p-metric [4]. Notice that all these algorithms are *single-shot* algorithms, that is, they compute a clustering of given points (without queries) just once.

There have been some results on range-analysis query problems related to clustering queries. Brass et al. [8] presented data structures of finding extent measures: the width, area, or perimeter of the convex hull of $P \cap Q$. Arya et al. [6] studied data structures that support clustering queries on the length of the minimum spanning tree of $P \cap Q$. Various types of range-aggregate nearest neighbor queries have also been studied [23, 24].

Nekrich and Smid [22] considered approximate range-aggregate queries such as the diameter or radius of the smallest enclosing ball for points in d-dimensional space. Basically, their algorithm constructs a d-dimensional range tree as a data structure, in which each node stores a δ-coreset of points in its subtree (but not explicitly), and applies range-searching

query algorithms on the tree, where δ is a positive value. Their algorithm works for any aggregate function that can be approximated using a decomposable coreset including coresets for the k-median, k-means and k-center clusterings. In this case, the size of the data structure is $O(kn \log^d n/\delta^2)$, and the query algorithm computes a (k, δ)-coreset of size $O(k \log^{d-1} n/\delta^2)$. However, their algorithm uses k and δ in constructing the data structure for the clusterings, and therefore k and δ are fixed over range-clustering queries.

Very recently, Abrahamsen et al. [1] considered k-center range-clustering queries for n points in d-dimensional space. They presented a method, for a query consisting of a range Q, an integer k with $1 \leq k \leq n$ and a value $\varepsilon > 0$, of computing a (k, ε)-coreset S from $P \cap Q$ of size $O(k/\varepsilon^d)$ in $O(k(\log n/\varepsilon)^{d-1} + k/\varepsilon^d)$ time such that the k-center of S is an $(1 + \varepsilon)$-approximation to the k-center of $P \cap Q$. After computing the coreset, they compute an $(1 + \varepsilon)$-approximation to the k-center of the coreset using a known single-shot algorithm. Their data structure is of size $O(n \log^{d-1} n)$ and its query algorithm computes an $(1+\varepsilon)$-approximate to a k-center range-clustering query in $O(k(\log n/\varepsilon)^{d-1} + T_{\text{ss}}(k/\varepsilon^d))$ time, where $T_{\text{ss}}(N)$ denotes the running time of an $(1 + \varepsilon)$-approximation single-shot algorithm for the k-center problem on N points.

The problem of computing the diameter of input points contained in a query range can be considered as a special case of the range-clustering problem. Gupta et al. [15] considered this problem in the plane and presented two data structures. One requires $O(n \log^2 n)$ size that supports queries with arbitrary approximation factors $1 + \varepsilon$ in $O(\log n/\sqrt{\varepsilon} + \log^3 n)$ query time and the other requires a smaller size $O(n \log n/\sqrt{\delta})$ that supports only queries with the *fixed* approximation factor $1 + \delta$ with $0 < \delta < 1$ that is used for constructing the data structure. The query time for the second data structure is $O(\log^3 n/\sqrt{\delta})$. Nekrich and Smid [22] presented a data structure for this problem in a higher dimensional space that has size $O(n \log^d n)$ and supports diameter queries with the fixed approximation factor $1 + \delta$ in $O(\log^{d-1} n/\delta^{d-1})$ query time. Here, δ is an approximation factor given for the construction of their data structure, and therefore it is fixed for queries to the data structure.

Our results. We present algorithms for k-median, k-means, and k-center range-clustering queries with query times polynomial to $\log n$, k and $1/\varepsilon$. These algorithms have a similar structure: they compute a (k, ε)-coreset of input points contained in a query range, and then compute a clustering on the coreset using a known clustering algorithm. We use $T_{\text{ss}}(N)$ to denotes the running time for any $(1+\varepsilon)$-approximation single-shot algorithm of each problem on N points. For a set P of n points in $\mathbb{R}^d$, we present the following results.

- There is a data structure of size $O(n \log^d n)$ such that an $(1 + \varepsilon)$-approximation to the k-median or k-means clustering of $P \cap Q$ can be computed in time

  $$O(k^5 \log^9 n + k \log^d n/\varepsilon + T_{\text{ss}}(k \log n/\varepsilon^d))$$

  for any box Q, any integer k with $1 \leq k \leq n$, and any value $\varepsilon > 0$ given as a query.
- There is a data structure of size $O(n \log^{d-1} n)$ such that an $(1 + \varepsilon)$-approximation to the k-center clustering of $P \cap Q$ can be computed in time

  $$O(k \log^{d-1} n + k \log n/\varepsilon^{d-1} + T_{\text{ss}}(k/\varepsilon^d))$$

  for any box Q, an integer k with $1 \leq k \leq n$, and a value $\varepsilon > 0$ given as a query.
- There is a data structure of size $O(n \log^{d-1} n)$ such that an $(1 + \varepsilon)$-approximation to the diameter (or radius) of $P \cap Q$ can be computed in time

  $$O(\log^{d-1} n + \log n/\varepsilon^{d-1})$$

  for any box Q and a value $\varepsilon > 0$ given as a query.

Our results are obtained by combining range searching with coresets. The k-median and k-means range-clusterings have not been studied before, except the work by Nekrich and Smid. They presented a general method to compute approximate range-aggregate queries, which can be used for the k-median or k-means range-clustering. Here, the approximation factor is required to be given in the construction of their data structure as mentioned above. However, it is not clear how to use or make their data structure to support approximate range-clustering queries with various approximation factors we consider in this paper. Indeed, the full version of the paper by Abrahamsen et al. [1] poses as an open problem a data structure supporting $(1+\varepsilon)$-approximate k-median or k-means range-clustering queries with various values ε. Our first result answers to the question and presents a data structure for the k-median and k-means range-clustering problems for any value ε.

Our second result improves the best known previous work by Abrahamsen et al. [1]. Recall that the query algorithm by Abrahamsen et al. takes $O(k(\log n/\varepsilon)^{d-1} + T_{\mathrm{ss}}(k/\varepsilon^d))$ time. We improved the first term of their running time by a factor of $\min\{1/\varepsilon^{d-1}, \log^{d-2} n\}$.

Our third result improves the best known previous work by Nekrich and Smid [22]. Our third result not only allows queries to have arbitrary approximation factor values $1+\varepsilon$, but also improves the size and the query time of these data structures. The size is improved by a factor of $\log n$. Even when ε is fixed to δ, the query time is improved by a factor of $\min\{1/\delta^{d-1}, \log^{d-2} n\}$ compared to the one by Nekrich and Smid [22].

A main tool achieving the three results is a new data structure for range-emptiness and range-counting queries. Consider a grid Γ with side length γ covering an axis-parallel hypercube with side length ℓ that is aligned with the standard quadtree. For a given query range Q and every cell $\square$ of Γ, we want to check whether there is a point of P contained in $\square \cap Q$ efficiently. (Or, we want to compute the number of points of P contained in $\square \cap Q$.) For this purpose, one can use a data structure for orthogonal range-emptiness queries supporting $O(\log^{d-1} n)$ query time [10]. Thus, the task takes $O((\ell/\gamma)^d \log^{d-1} n)$ time for all cells of Γ in total. Notice that $(\ell/\gamma)^d$ is the number of grid cells of Γ.

To improve the running time for the task, we present a new data structure that supports a range-emptiness query in $O(\log^{d-t-1} n + \log n)$ time for a cell of Γ intersecting no face of Q with dimension smaller than t for any fixed t. Using our data structure, the running time for the task is improved to $O(\log^{d-1} n + (\ell/\gamma)^d \log n)$. To obtain this data structure, we observe that a range-emptiness query for $\square \cap Q$ can be reduced to a $(d-t-1)$-dimensional orthogonal range-emptiness query on points contained in $\square$ if $\square$ intersects no face of Q with dimension smaller than t. We maintain a data structure for $(d-t-1)$-dimensional orthogonal range-emptiness queries for each cell of predetermined grids, which we call the compressed quadtree, for every t. However, this requires $\Omega(n^2)$ space in total if we maintain these data structures explicitly. We can reduce the space complexity using a method for making a data structure partially persistent [11].

Another tool to achieve an efficient query time is a *unified grid* based on quadtrees. For the k-median and k-means clusterings, we mainly follow an approach given by Har-Peled and Mazumdar [18]. They partition input points with respect to the approximate centers explicitly, and construct a grid for each subset of the partition. They snap each input point p to a cell of the grid constructed for the subset that p belongs to. However, their algorithm is a single-shot algorithm and requires $\Omega(|P \cap Q|)$ time due to the computation of a coreset from approximate centers of the points contained in a given query box. In our algorithm, we do not partition input points explicitly but we use only one grid instead, which we call the unified grid, for the purpose in the implementation so that a coreset can be constructed more efficiently.

The tools we propose in this paper to implement the algorithm by Har-Peled and Mazumdar can be used for implementing other algorithms based on grids. For example, if Monte Carlo algorithms are allowed, the approach given by Chen [9] for approximate range queries can be implemented by using the tools we propose in this paper.

Due to lack of space, some proofs and details are omitted. The missing proofs and details can be found in the full version of the paper.

2 Preliminaries

For any two points x and y in $\mathbb{R}^d$, we use $d(x, y)$ to denote the Euclidean distance between x and y. For a point x and a set Y in $\mathbb{R}^d$, we use $d(x, Y)$ to denote the smallest Euclidean distance between x and any point in Y. Throughout the paper, we use the terms *square* and *rectangle* in a generic way to refer axis-parallel hypercube and box in $\mathbb{R}^d$, respectively.

Clusterings. For any integer k with $1 \leq k \leq n$, let $\mathcal{C}_k$ be the family of the sets of at most k points in $\mathbb{R}^d$. Let $\Phi : \mathcal{C}_n \times \mathcal{C}_k \to \mathbb{R}_{\geq 0}$ be a cost function which will be defined later. For a set $P \subseteq \mathbb{R}^d$, we define $\textsc{Opt}_k(P)$ as the minimum value $\Phi(P, C)$ over all sets $C \in \mathcal{C}_k$. We call a set $C \in \mathcal{C}_k$ realizing $\textsc{Opt}_k(P)$ a *k-clustering of P under the cost function Φ*.

In this paper, we consider three cost functions $\Phi_{\mathsf{M}}, \Phi_{\mathsf{m}}$ and Φ_{c} that define the k-median, k-means, and k-center clusterings, respectively. Let $\phi(p, C) = \min_{c \in C} d(p, c)$ for any point p in P. The cost functions are defined as follows: for any set C of $\mathcal{C}_k$,

$$\Phi_{\mathsf{M}}(P, C) = \sum_{p \in P} \phi(p, C), \quad \Phi_{\mathsf{m}}(P, C) = \sum_{p \in P} (\phi(p, C))^2, \quad \Phi_{\mathsf{c}}(P, C) = \max_{p \in P} \phi(p, C).$$

We consider the query variants of these problems. We preprocess P so that given a query rectangle Q, an integer k with $1 \leq k \leq n$, and a value $\varepsilon > 0$, an $(1 + \varepsilon)$-approximate k-clustering of the points contained in Q can be reported efficiently. Specifically, we want to report a set $C \in \mathcal{C}_k$ with $\Phi(P_Q, C) \leq (1 + \varepsilon)\textsc{Opt}_k(P_Q)$ in sublinear time, where $P_Q = P \cap Q$.

Coreset for clustering. Consider a cost function Φ. We call a set $S \subseteq \mathbb{R}^d$ a (k, ε)-*coreset* of P for the k-clustering under the cost function Φ if the following holds: for any set C of $\mathcal{C}_k$,

$$(1 - \varepsilon)\Phi(P, C) \leq \Phi(S, C) \leq (1 + \varepsilon)\Phi(P, C).$$

Here, the points in a coreset might be weighted. In this case, the distance between a point p in d-dimensional space and a weighted point s in a coreset is defined as $w(s) \cdot d(p, s)$, where $w(s)$ is the weight of s and $d(p, s)$ is the Euclidean distance between p and s.

By definition, an $(1 + \varepsilon)$-approximation to the k-clustering of S is also an $(1 + \varepsilon)$-approximation to the k-clustering of P. Thus, (k, ε)-coresets can be used to obtain a fast approximation algorithm for the k-median, k-means, and k-center clusterings. A (k, ε)-coreset of smaller size gives a faster approximation algorithm for the clusterings. The followings are the sizes of the smallest (k, ε)-coresets known so far: $O(k/\varepsilon^2)$ for the k-median and k-means clusterings in $\mathbb{R}^d$ [13], and $O(k/\varepsilon^d)$ for the k-center clustering in $\mathbb{R}^d$ [3].

It is also known that (k, ε)-coresets for the k-median, k-means, and k-center clusterings are *decomposable*. That is, if S_1 and S_2 are (k, ε)-coresets for disjoint sets P_1 and P_2, respectively, then $S_1 \cup S_2$ is a (k, ε)-coreset for $P_1 \cup P_2$ by [18, Observation 7.1]. Using this property, one can obtain data structures on P that support an $(1 + \delta)$-approximation to the k-median, k-means, and k-center range-clustering queries for constants $\delta > 0$ and k with $1 \leq k \leq n$ which are given in the construction phase.

▶ **Lemma 1.** *Given a set P of n points in $\mathbb{R}^d$ and a value $\delta > 0$ given in the construction phase, we can construct a data structure of size $O(n \log^d n)$ so that a (k, δ)-coreset of $P \cap Q$ for the k-median and k-means clusterings of size $O(k \log^d n)$ can be computed in $O(k \log^d n)$ time for any orthogonal range Q and any integer k with $1 \leq k \leq n$ given as a query.*

Note that the approximation factor of the coreset is fixed to queries. In Section 4, we will describe a data structure and its corresponding query algorithm answering k-median and k-means range-clustering queries that allow queries to have arbitrary approximation factor values $1 + \varepsilon$ using the algorithm in Lemma 1 as a subprocedure.

Single-shot algorithms for the k-median and k-means clusterings. The single-shot version of this problem was studied by Har-Peled and Mazumdar [18]. They gave algorithms to compute (k, ε)-coresets of size $O(k \log n/\varepsilon^d)$ for the k-median and k-means clusterings. We extensively use their results. The algorithm for the k-means clustering works similarly.

They first provide a procedure that given a constant-factor approximation A to the k-means clustering of P consisting of possibly more than k centers, computes a (k, ε)-coreset of P for the k-median clustering of size $O(|A| \log n/\varepsilon^d)$. To do this, the procedure partitions P into m pairwise disjoint subsets with respect to A and constructs a grid for each subset with respect to the approximate centers. For every grid cell $\square$ containing a point of a subset P' of the partition, the procedure picks an arbitrary point q of P' contained in it and assigns the number of points of P' contained in $\square$ to q as its weight. They showed that the set of all weighted points is a (k, ε)-coreset for P of size $O(|A| \log n/\varepsilon^d)$.

Using this procedure, the algorithm constructs a (k, ε)-coreset S of size $O(k \log^4 n/\varepsilon^d)$ using a constant-factor approximation of size $O(k \log^3 n)$. Using the coreset S, the algorithm obtains a smaller coreset of size $O(k \log n/\varepsilon^d)$ as follows. The algorithm computes a constant-factor approximation $\mathcal{C}_0$ to the k-center clustering of S using the algorithm in [14]. This clustering is an $O(n)$-approximation to the k-median clustering. Then it applies the local search algorithm by Arya et al. [7] to $\mathcal{C}_0$ and S to obtain a constant-factor approximation of P of size at most k. It uses this set to compute a (k, ε)-coreset of size $O(k \log n/\varepsilon^d)$ by applying the procedure mentioned above again.

3 Data structures for range-clustering queries

We maintain two data structures constructed on P. One is a compressed quadtree [5], and the other is a variant of a range tree, which we introduce in this paper.

3.1 Compressed quadtree

We use the term *quadtrees* in a generic way to refer the hierarchical spatial tree data structures for d-dimensional data that are based on the principle of recursive decomposition of space, also known as quadtrees, octrees, and hyperoctrees for spatial data in $d = 2, 3$, and higher dimensions, respectively. We consider two types of quadtrees: a standard quadtree and a compressed quadtree.

Without loss of generality, we assume that the side length of the square corresponding to the root is 1. Then every cell of the standard quadtree has side length of 2^{-i} for an integer i with $0 \leq i \leq t$ for some constant t. We call a value of form 2^{-i} for an integer i with $0 \leq i \leq t$ a *standard length*. Also, we call a grid a *standard grid* if every cell of the grid is also in the standard quadtree. In other words, any grid aligned with the standard

quadtree is called a standard grid. We use $\mathcal{T}_s$ and $\mathcal{T}_c$ to denote the standard and compressed quadtrees constructed on P, respectively.

For ease of description, we will first introduce our algorithm in terms of the standard quadtree. But the algorithm will be implemented using the compressed quadtree to reduce the space complexity. To do this, we need the following lemma. In the following, we use a node and its corresponding cell of $\mathcal{T}_s$ (and $\mathcal{T}_c$) interchangeably. For a cell $\square$ of the standard quadtree on P, there is a unique cell $\overline{\square}$ of the compressed quadtree on P satisfying $\square \cap P = \overline{\square} \cap P$. We call this cell the *compressed cell* of $\square$.

▶ **Lemma 2** ([16]). *We can find the compressed cell of a given cell $\square$ of $\mathcal{T}_s$ in $O(\log n)$ time.*

3.2 Data structure for range-emptiness queries

In our query algorithm, we consider a standard grid Γ of side length γ covering an axis-parallel hypercube of side length ℓ. For a given query range Q and every cell $\square$ of Γ, we want to check whether there is a point of P contained in $\square \cap Q$ efficiently. For this purpose, one can use a data structure for orthogonal range-emptiness queries supporting $O(\log^{d-1} n)$ query time [10]. Thus, the task takes $O((\ell/\gamma)^d \log^{d-1} n)$ time for all cells of Γ in total. Notice that $(\ell/\gamma)^d$ is the number of grid cells of Γ.

However, we can accomplish this task more efficiently using the data structure which we will introduce in this section. Let t be an integer with $0 < t \leq d$. We use $<_t$-face to denote a face with dimension smaller than t among faces of a rectangle in $\mathbb{R}^d$. Note that a corner of a rectangle in $\mathbb{R}^d$ is a $<_t$-face of a rectangle for $t = 1$. Our data structure allows us to check whether a point of P is contained in $Q \cap \overline{\square}$ in $O(\log^{d-t-1} n + \log n)$ time for a cell $\overline{\square}$ of $\mathcal{T}_c$ intersecting no $<_t$-face of Q with $0 < t < d$. Here, we first compute the compressed cell $\overline{\square}$ of each cell $\square$ in Γ, and then apply the query algorithm to $\overline{\square}$. Recall that $\square \cap P$ coincides with $\overline{\square} \cap P$ for any cell $\square$ of Γ and its compressed cell $\overline{\square}$. In this way, we can complete the task in $O(\sum_{t=1}^{d-1} x_t \log^{d-t-1} n + |\Gamma| \log n + \log^{d-1} n)$ time in total, where x_t is the number of cells of Γ intersecting no $<_t$-face of Q but intersecting a t-dimensional face of Q. Notice that for any cell $\square$ intersecting Q, there is an integer t with $0 < t \leq d$ such that $\square$ intersects no $<_t$-face of Q, but intersects a t-dimensional face of Q unless $\square$ contains a corner of Q. Here, x_t is $O((\ell/\gamma)^t)$. Therefore, we can accomplish the task for every cell of Γ in $O(\log^{d-1} n + (\ell/\gamma)^d \log n)$ time in total.

For a nonempty subset I of $\{1, \ldots, d\}$ of size t, the *I-projection range tree* on a point set $A \subseteq \mathbb{R}^d$ is the range tree supporting fractional cascading constructed on the projections of A onto a $(d-t)$-dimensional hyperplane orthogonal to the ith axes for all $i \in I$.

▶ **Lemma 3.** *Given the I-projection range tree on $P \cap \square$ for every cell $\square$ of $\mathcal{T}_c$ and every nonempty subset I of $\{1, \ldots, d\}$, we can check whether a point of P is contained in $Q \cap \square$ for any query rectangle Q and any cell $\square$ of $\mathcal{T}_c$ intersecting no $<_t$-face of Q in $O(\log^{d-t-1} n + \log n)$ time.*

However, the I-projection range trees require $\Omega(n^2)$ space in total if we store them explicitly. To reduce the space complexity, we use a method of making a data structure *partially persistent* [11]. A partially persistent data structure allows us to access any elements of an old version of the data structure by keeping the changes on the data structure. Driscoll et al. [11] presented a general method of making a data structure based on pointers partially persistent. In their method, both time and space overheads for an update are $O(1)$ amortized, and the access time for any version is $O(\log n)$.

3.2.1 Construction of the I-projection range trees

Consider a fixed subset I of $\{1, \ldots, d\}$. We construct the I-projection range trees for the cells of $\mathcal{T}_c$ in a bottom-up fashion, from leaf cells to the root cell. Note that each leaf cell of the compressed quadtree contains at most one point of P. We initially construct the I-projection range tree for each leaf cell of $\mathcal{T}_c$ in total $O(n)$ time.

Assume that we already have the I-projection range tree for every child node of an internal node v with cell $\square$ of $\mathcal{T}_c$. Note that an internal node of the compressed quadtree has at least two child nodes and up to 2^d child nodes. We are going to construct the I-projection range tree for v from the I-projection range trees of the child nodes of v. One may consider to merge the I-projection range trees for the child nodes of v into one, but we do not know any efficient way of doing it. Instead, we construct the I-projection range tree for v as follows. Let u be a child node of v on $\mathcal{T}_c$ that contains the largest number of points of P in its corresponding cell among all child nodes of v. We insert all points of P contained in the cells for the child nodes of v other than u to the I-projection range tree of u to form the I-projection range tree of v. Here, we do not destroy the old version of the I-projection range tree of u by using the method by Driscoll et al. [11]. Therefore, we can still access the I-projection range tree of u. For the insertion, we use an algorithm that allows us to apply fractional cascading on the range tree under insertions of points [21]. We do this for every subset I of $\{1, \ldots, d\}$ and construct the I-projection range trees for nodes of $\mathcal{T}_c$.

In this way, we can access any I-projection range tree in $O(\log n)$ time, and therefore we can check if a point of P is contained in $Q \cap \square$ in $O(\log^{d-t-1} n + \log n)$ time for any rectangle Q and any cell $\square$ of $\mathcal{T}_c$ intersecting no $<_t$-face of Q for any integer t with $0 < t \leq d$ by Lemma 3.

3.2.2 Analysis of the construction

The construction of the dynamic range tree [21, Theorem 8] requires $O(\delta \log^{d-t-1} \delta)$ space on the insertions of δ points in $\mathbb{R}^{d-t}$. The method by Driscoll et al. requires only $O(1)$ overhead for each insertion on the space complexity. Thus, the space complexity of the I-projection range trees for a fixed subset I consisting of t indices over all cells of $\mathcal{T}_c$ is $O(n + \delta \log^{d-t-1} \delta)$, where δ is the number of the insertions performed during the construction in total.

The update procedure for the dynamic range tree [21, Theorem 8] takes $O(\log^{d-t-1} n)$ time if only insertions are allowed. The method by Driscoll requires only $O(1)$ overhead for each insertion on the update time. Thus, the construction time is $O(n + \delta \log^{d-t-1} n)$, where δ is the number of the insertions performed during the construction in total.

The following lemma shows that δ is $O(n \log n)$, and thus our construction time is $O(n \log^{d-t} n)$ and the space complexity of the data structure is $O(n \log^{d-t} n)$ for each integer t with $0 < t \leq d$. Note that there are $2^d = O(1)$ subsets of $\{1, \ldots, d\}$. Therefore, the total space complexity and construction time are $O(n \log^{d-1} n)$.

▶ **Lemma 4.** *For a fixed subset I of $\{1, \ldots, d\}$, the total number of insertions performed during the construction of all I-projection range trees for every node of $\mathcal{T}_c$ is $O(n \log n)$.*

▶ **Lemma 5.** *We can construct a data structure of size $O(n \log^{d-1} n)$ in $O(n \log^{d-1} n)$ time so that the emptiness of $P \cap Q \cap \square$ can be checked in $O(\log^{d-t-1} n + \log n)$ for any query rectangle Q and any cell $\square$ of $\mathcal{T}_c$ intersecting no $<_t$-face of Q for an integer t with $0 < t \leq d$.*

For a cell $\square$ containing a corner of Q, there is no index t such that $\square$ intersects no $<_t$-face of Q. In this case, we use the standard range tree on P and check the emptiness of $P \cap Q \cap \square$ in $O(\log^{d-1} n)$ time. Notice that there are 2^d such cells because the cells are pairwise disjoint.

3.3 Data structure for range-counting queries

The data structure for range-emptiness queries described in Section 3.2 can be extended to a data structure for range-reporting queries. However, it does not seem to work for range-counting queries. This is because the dynamic range tree with fractional cascading [21] does not seem to support counting queries. Instead, we use a dynamic range tree without fractional cascading, which increases the query time and update time by a factor of $\log n$. The other part is the same as the data structure for range-emptiness queries.

▶ **Lemma 6.** *We can construct a data structure of size $O(n \log^{d-1} n)$ in $O(n \log^{d-1} n)$ time so that the number of points of P contained in $Q \cap \square$ can be computed in $O(\log^{d-t} n + \log n)$ time for any query rectangle Q and any cell $\square$ of $\mathcal{T}_c$ intersecting no $<_t$-face of Q for an integer t with $0 < t \leq d$.*

4 k-Median range-clustering queries

In this section, we present a data structure and a query algorithm for k-median range-clustering queries. Given a set P of n points in $\mathbb{R}^d$ for a constant $d \geq 2$, our goal is to preprocess P such that k-median range-clustering queries can be answered efficiently. A k-median range-clustering query consists of a box Q, an integer k with $1 \leq k \leq n$, and a value $\varepsilon > 0$. We want to find a set $C \in \mathcal{C}_k$ with $\Phi_{\mathsf{M}}(P_Q, C) \leq (1+\varepsilon)\textsc{Opt}_k(P_Q)$ efficiently, where $P_Q = P \cap Q$. Throughout this section, we use Φ to denote Φ_{M} unless otherwise specified.

Our query algorithm is based on the single-shot algorithm by Har-Peled and Mazumdar [18]. A main difficulty in the implementation for our setting is that they construct a grid with respect to each point in an approximate center set. Then for each grid cell, they compute the number of points of P_Q contained in the grid cell. Thus to implement their approach in our setting directly, we need to apply a counting query to each grid cell. Moreover, we have to avoid overcounting as a point might be contained in more than one grid cell of their grid structures.

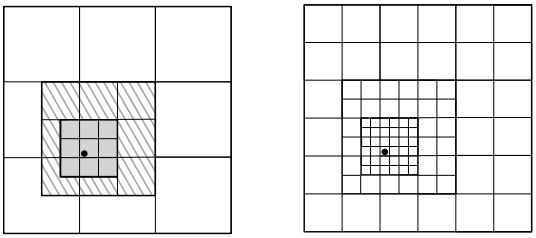
To do this efficiently without overcounting, we use a *unified grid* based on the standard quadtree. Although this grid is defined on the standard quadtree, we use the grid on the compressed quadtree in the implementation To make the description easier, we use the standard quadtree instead of the compressed quadtree in defining of the unified grid.

4.1 Coreset construction from approximate centers

Assume that we are given a constant-factor approximation $A = \{a_1, \ldots, a_m\}$ to the k-median clustering of P_Q, where m is possibly larger than k. In this subsection, we present a procedure that computes a (k, ε)-coreset of size $O(|A| \log n/\varepsilon^d)$. We follow the approach by Har-Peled and Mazumdar [18] and implement it in our setting.

General strategy. We describe our general strategy first, and then show how to implement this algorithm. For the definitions of the notations used in the following, refer to those in Section 2 unless they are given. We compute a $2\sqrt{d}$-approximation R to the maximum of $d(p, A)/(c_1|P_Q|)$ over all points $p \in P_Q$, that is, a value R satisfying that the maximum value lies between $R/(2\sqrt{d})$ and $2\sqrt{d}R$, where $c_1 > 1$ is the approximation factor of A.

Let Q_{ij} be the cell of the standard quadtree containing a_i with side length $\bar{R}_j$ satisfying $R2^j \leq \bar{R}_j < R2^{j+1}$ for $j = 0, \ldots, M = \lceil 2\log(2\sqrt{d}c_1|P_Q|)\rceil$. By construction, note that $Q_{ij_1} \subset Q_{ij_2}$ for any two indices j_1 and j_2 with $j_1 < j_2$. For any point p in P_Q, we have at least one cell Q_{ij} containing p since there is a value $\bar{R}_j$ at least four times the maximum of $d(p, A)$.



(a) (b)

Figure 1 (a) Exponential grid for an algorithm for the query version constructed with respect to the point, say a_1, in the middle. The point is not the center of the grid. The gray region is the grid cluster for Q_{10}, the dashed region is the grid cluster for Q_{11}, and the largest box is the grid cluster for Q_{12}. (b) Only first-level grid is depicted.

We define the *grid cluster* for Q_{ij} as the union of at most 3^d grid cells of the standard quadtree with side length $\bar{R}_j$ that share their faces with Q_{ij} including Q_{ij}. Note that the grid cluster for Q_{ij} contains all points of $\mathbb{R}^d$ that are within distance from a_i at most $\bar{R}_j$. Also, every point in $\mathbb{R}^d$ contained in the grid cluster for Q_{ij} has its distance from a_i at most $2\sqrt{d}\bar{R}_j$. See Figure 1(a). Let V_{i0} denote the grid cluster for Q_{i0} and V_{ij} be the grid cluster for Q_{ij} excluding the grid cluster for $Q_{i(j-1)}$. Note that V_{ij} is the union of at most $3^d(2^d-1)$ cells of the standard quadtree with side length $\bar{R}_j/2$, except for $j=0$. For $j=0$, the region V_{i0} is the union of at most 3^d such cells.

The *first-level grid* for a fixed index i consists of all cells of the standard quadtree with side length $\bar{R}_j/2$ contained in V_{ij}. For an illustration, see Figure 1(b). We partition each cell of the first-level grid into the cells of the standard quadtree with side length $\bar{r}_j$ satisfying $\varepsilon\bar{R}_j/(40c_1d) \leq \bar{r}_j \leq 2\varepsilon\bar{R}_j/(40c_1d)$. The *second-level grid* for i consists of all such cells. Let $\mathcal{V}$ be the set of all grid cells which contain at least one point of P_Q. Note that the size of $\mathcal{V}$ is $O(|A|\log n/\varepsilon^d)$. We will compute it in $O(|A|\log^d n/\varepsilon + |A|\log n/\varepsilon^d)$ time.

We consider the grid cells $\square$ of $\mathcal{V}$ one by one in the increasing order of their side lengths, and do the followings. Let $P(\square)$ be the set of points of P_Q that are contained in $\square$, but are not contained in any other grid cells we have considered so far. We compute the number of points of $P(\square)$, and assign this number to an arbitrary point of $P(\square)$ as its weight. We call this weighted point the *representative* of $\square$. Also, we say that a point of $P(\square)$ is *charged* to $\square$. Notice that every point of P_Q is charged to exactly one cell of $\mathcal{V}$. We describe the details of this procedure in Section 4.1.2. Let S be the set of all such weighted points. Then S is a (k,ε)-coreset for P_Q of size $O(|A|\log n/\varepsilon^d)$.

We implement the algorithm using the compressed quadtree, not the standard quadtree. We provide an implementation of the algorithm in the following subsections briefly.

4.1.1 Computing an approximation to the average radius

The first step is to compute a $2\sqrt{d}$-approximation R to the maximum MAX of $d(p,A)/(c_1|P_Q|)$ over all points $p \in P_Q$, where $c_1 > 1$ is the approximation factor of A. More precisely, we compute R such that $R/(2\sqrt{d}) \leq \text{MAX} \leq 2\sqrt{d}R$. We can compute it in $O(|A|\log^d n)$ time.

Let r^* be the maximum of $d(p,A)$ over all points $p \in P_Q$. We compute a $2\sqrt{d}$-approximation of r^* and divide it by $c_1|P_Q|$ to compute R. Note that we can compute $|P_Q|$ in $O(\log^{d-1} n)$ time using the range tree constructed on P. Imagine that we have a standard grid with side length $\alpha > 0$ covering Q. Consider the grid cells in this grid each of which contains a point of A. If the union of the grid clusters of all these grid cells contains P_Q, it holds that $d(p,A) \leq 2\alpha\sqrt{d}$ for any $p \in P_Q$. Otherwise, $d(p,A) > \alpha$ for some $p \in P_Q$. We use this observation to check whether $2\alpha\sqrt{d} \geq r^*$ or $\alpha \leq r^*$.

We apply binary search on the standard lengths. For each iteration with standard length α, we check whether α is at most r^* or at least $r^*/(\alpha\sqrt{d})$. As a result, we obtain an interval whose endpoints are standard lengths, and return the larger endpoint as an output. However, there are an arbitrarily large number of distinct standard lengths. We consider only $O(\log n)$ distinct standard lengths among them.

▶ **Lemma 7.** *We can compute a $2\sqrt{d}$-approximation to the maximum of $d(p, A)/(c_1|P_Q|)$ for all points p in P_Q in $O(|A|\log^d n)$ time.*

4.1.2 Computing the compressed cells in the grid

As described in Section 4.1, we construct the second-level grid for each index i for $i = 1, \ldots, m$, and check whether each grid cell contains a point of P_Q. The set of the grid cells in the second-level grids containing a point of P_Q is denoted by $\mathcal{V}$. Then we consider the grid cells $\square$ of $\mathcal{V}$ one by one in the increasing order of their side lengths, and compute the number of points of P_Q contained in $\square$, but not contained in any other grid cells we have considered so far. Computing this number is quite tricky.

To handle this problem, we observe that for any two cells in $\mathcal{V}$, either they are disjoint or one is contained in the other because they are cells of the standard quadtree. For two cells $\square_1$ and $\square_2$ with $\square_1 \subseteq \square_2$, let i_1 and i_2 be the indices such that $\square_1$ and $\square_2$ are the grid cells of the second-level grids for i_1 and i_2, respectively. Since the grid cells in the same second-level grid are pairwise interior disjoint, we have $i_1 \neq i_2$. In this case, for any point $p \in \square_2$, there is another grid cell $\square_1'$ containing p in the second-level grid for i_1 with side length smaller than the side length of $\square_2$. Therefore, we do not consider any cell of $\mathcal{V}$ containing another cell of $\mathcal{V}$. Imagine that we remove all such cells from $\mathcal{V}$. Then the cells of $\mathcal{V}$ are pairwise interior disjoint. Therefore, if suffices to compute the number of points of P_Q contained in each cell of $\mathcal{V}$, which can be done efficiently using the data structure in Section 3.

We show how to compute the set $\mathcal{V}$ after removing all cells containing another cell efficiently. To do this, we first compute the cells in the first-level grids, and discard some of them. Then we subdivide the remaining cells into cells in the second-level grids. More specifically, let $\mathcal{V}_1$ be the set of the cells of the first-level grids. We first compute the cells in $\mathcal{V}_1$, and then remove all cells in $\mathcal{V}_1$ containing another cell in $\mathcal{V}_1$. Then the cells in $\mathcal{V}_1$ are pairwise interior disjoint. And then we compute the second-level grid cells in each cell of $\mathcal{V}_1$. The second-level grid cells we obtain are the cells of $\mathcal{V}$ containing no other cell in $\mathcal{V}$.

Range-counting for each compressed cell. The next step is to compute the number of points of P_Q contained in each cell $\square$ of $\mathcal{V}$. If $\square$ is contained in Q, we already have the number of points of P_Q contained in $\square$, which is computed in the preprocessing phase. If $\square$ contains a corner of Q, we use the range tree constructed on P. Since there are $O(1)$ such cells, we can handle them in $O(\log^{d-1} n)$ time in total. For the remaining cells, we use the data structure in Section 3.3. Then we can handle them in $O(\sum_{t=1}^{d-1} m_t \log^{d-t} n)$ time, where m_t is the number of the cells of $\mathcal{V}$ intersecting no $<_t$-face of Q but intersecting a t-dimensional face of Q for an integer with $0 < t < d$. We have $m_t = O(|A|\log n/\varepsilon^t)$. Therefore, the total running time for the range-counting queries is $O(|A|\log^2 n/\varepsilon^{d-1} + |A|\log^d n/\varepsilon + \log^{d-1} n + |A|\log n/\varepsilon^d)$ in total, which is $O(|A|\log^d n/\varepsilon + |A|\log n/\varepsilon^d)$.

▶ **Lemma 8.** *Given a constant-factor approximation A to the k-median clustering of a set P of n points in $\mathbb{R}^d$ such that $|A|$ is possibly larger than k, we can compute a (k, ε)-coreset of P_Q of size $O(|A|\log n/\varepsilon^d)$ in $O(|A|\log^d n/\varepsilon + |A|\log n/\varepsilon^d)$ time for any rectangle Q, any integer k with $1 \leq k \leq n$ and any value $\varepsilon > 0$.*

4.2 Smaller coreset

Due to Lemma 1, we can obtain a $(k, 2)$-coreset S of P_Q of size $O(k \log^d n)$ in $O(k \log^d n)$ time for any query rectangle Q using a data structure of size $O(n \log^d n)$. A (k, c)-coreset of S is also a $(k, 2c)$-coreset of P_Q for any constant $c > 1$ by the definition of the coreset. We compute a $(k, 2)$-corset S' of S, which is a $(k, 4)$-coreset of P_Q, of size $O(k \log n)$ in $O(k \log^d n + k^5 \log^9 n)$ time by [18, Lemma 5.1] by setting $\varepsilon = 2$.

Using this $(k, 4)$-coreset of size $O(k \log n)$ of P_Q, we can obtain constant-factor approximate centers of size k as Har-Peled and Mazumdar [18] do. Then we can compute a (k, ε)-coreset of size $O(k \log n/\varepsilon^d)$ using Lemma 8 again using the constant-factor approximation centers to $\textsc{Opt}_k(S)$ of size k.

This algorithm is extended to the k-means range-clustering problem.

▶ **Theorem 9.** *There is a data structure of size $O(n \log^d n)$ on a set P of n points such that given a box $Q \subseteq \mathbb{R}^d$, an integer k with $1 \leq k \leq n$, and a value $\varepsilon > 0$ as a query, an $(1 + \varepsilon)$-approximation to the k-median range-clustering of $P \cap Q$ can be computed in $O(k^5 \log^9 n + k \log^d /\varepsilon + T_{\text{ss}}(k \log n/\varepsilon^d))$ time, where $T_{\text{ss}}(N)$ denotes the running time of an $(1 + \varepsilon)$-approximation single-shot algorithm for the k-median clustering of N weighted points.*

Remark. The construction of the coreset for the k-means clustering is similar to the construction of the coreset for the k-median clustering in [18]. The only difference is that for the k-means clustering Φ_{m} is used instead of Φ_{M} and $R = \sqrt{\Phi_{\mathsf{m}}(P, A)/(c_1 n)}$ is used instead of $R = \Phi_{\mathsf{M}}(P, A)/(c_1 n)$. Therefore, we can compute a (k, ε)-coreset for the k-means clustering of size $O(k \log n/\varepsilon^d)$ in $O(k^5 \log^9 n + k \log^d n/\varepsilon + k \log n/\varepsilon^d)$ time.

▶ **Theorem 10.** *Let P be a set of n points in d-dimensional space. There is a data structure of size $O(n \log^d n)$ such that given a query range $Q \subseteq \mathbb{R}^d$, an integer k with $1 \leq k \leq n$, and a value $\varepsilon > 0$ as a query, an $(1 + \varepsilon)$-approximation to the k-means range-clustering of $P \cap Q$ can be computed in $O(k^5 \log^9 n + k \log^d /\varepsilon + T_{\text{ss}}(k \log n/\varepsilon^d))$ time, where $T_{\text{ss}}(N)$ denotes the running time of an $(1 + \varepsilon)$-approximation single-shot algorithm for the k-means clustering of N weighted input points.*

5 k-Center range-clustering queries

We are given a set P of n points in $\mathbb{R}^d$ for a constant $d \geq 2$. Our goal is to process P so that k-center range-clustering queries can be computed efficiently. A range-clustering query consists of a rectangle $Q \subseteq \mathbb{R}^d$, an integer k with $1 \leq k \leq n$, and a value $\varepsilon > 0$. We want to find a set $C \in \mathcal{C}_k$ with $\Phi_{\mathsf{c}}(P_Q, C) \leq (1 + \varepsilon)\textsc{Opt}_k(P_Q)$ efficiently, where $P_Q = P \cap Q$. Combining the algorithm by Abrahamsen et al. [1] and the data structure in Section 3, we can improve the algorithm by Abrahamsen et al.

▶ **Theorem 11.** *There is a data structure of size $O(n \log^{d-1} n)$ on a set P of n points in $\mathbb{R}^d$ such that given a box $Q \subseteq \mathbb{R}^d$, an integer k with $1 \leq k \leq n$, and a value $\varepsilon > 0$ as a query, an $(1 + \varepsilon)$-approximation to the k-center range-clustering of $P \cap Q$ can be computed in $O(k \log^{d-1} n + k \log n/\varepsilon^{d-1} + T_{\text{ss}}(k/\varepsilon^d))$ time, where $T_{\text{ss}}(N)$ denotes the running time of an $(1 + \varepsilon)$-approximation single-shot algorithm for the k-center clustering of N points.*

6 Approximate diameter and radius of a point set

In this section, we are given a set P of n points in $\mathbb{R}^d$. Our goal in this section is to preprocess P so that given any orthogonal range Q and a value $\varepsilon > 0$, an approximate diameter (or radius) of $P \cap Q$ can be computed efficiently. We give a sketch of the algorithm.

Our query algorithm starts by sampling a set S of points from $P \cap Q$, which we call an ε-coreset of $P \cap Q$, such that the diameter of S is an $(1+\varepsilon)$-approximation of the diameter of $P \cap Q$. Let APX be a value such that $D \leq \text{APX} \leq c \cdot D$ for a constant $c > 1$, where D is the diameter of $P \cap Q$. Consider a standard grid of side length εAPX covering Q. Assume that we pick an arbitrary point in each grid cell containing a point of $P \cap Q$. Then the set of all picked points is an ε-coreset of $P \cap Q$ of size $O(1/\varepsilon^d)$.

Let $\mathcal{D}$ be the set of all grid cells containing a point of $P \cap Q$. We can obtain a smaller ε-coreset as follows. We first obtain a subset $\mathcal{D}' \subseteq \mathcal{D}$ and choose an arbitrary point in each grid cell of $\mathcal{D}'$ for a ε-coreset. If a grid cell of $\mathcal{D}$ intersects the boundary of Q, we move it from $\mathcal{D}$ to $\mathcal{D}'$. For the remaining cells of $\mathcal{D}$, consider the grid cells of $\mathcal{D}$ whose centers have the same coordinates, except for only one coordinate, say the ith coordinate. We add the grid cells with the largest ith coordinate and smallest ith coordinate to $\mathcal{D}'$. Then $\mathcal{D}'$ consists of $O(1/\varepsilon^{d-1})$ grid cells. We choose an arbitrary point of P contained in each grid cell of $\mathcal{D}'$. The set S of all chosen points is an ε-coreset of $P \cap Q$ of size $O(1/\varepsilon^{d-1})$. Using the data structure described in Section 3, we can compute this coreset in $O(\log^{d-1} n + \log n/\varepsilon^{d-1})$ time. Also, this approach works for the problem of computing the radius of the smallest enclosing ball of $P \cap Q$.

▶ **Theorem 12.** *Given a set P of n points in $\mathbb{R}^d$, we can compute an $(1+\varepsilon)$-approximate diameter (or radius) of $P \cap Q$ in $O(\log^{d-1} n + \log n/\varepsilon^{d-1})$ time for a query consisting of an orthogonal range Q and a value $\varepsilon > 0$ using a data structure of size $O(n \log^{d-1} n)$.*

References

1. Mikkel Abrahamsen, Mark de Berg, Kevin Buchin, Mehran Mehr, and Ali D. Mehrabi. Range-Clustering Queries. In *Proceedings of the 33rd International Symposium on Computational Geometry (SoCG 2017)*, volume 77, pages 5:1–5:16, 2017.
2. Pankaj K. Agarwal and Jeff Erickson. Geometric range searching and its relatives. In *Advances in Discrete and Compputational Geometry*, volume 223 of *Contemporary Mathematics*, pages 1–56. American Mathematical Society Press, 1999.
3. Pankaj K. Agarwal, Sariel Har-Peled, and Kasturi R. Varadarajan. Geometric approximation via coresets. In *Combinatorial and Computational Geometry*, volume 52, pages 1–30. MSRI Publications, 2005.
4. Pankaj. K. Agarwal and Cecillia M. Procopiuc. Exact and approximation algorithms for clustering. *Algorithmica*, 33(2):201–226, 2002.
5. Srinivas Aluru. Quadtrees and octrees. In Dinesh P. Mehta and Sartaj Sahni, editors, *Handbook of Data Structures and Applications*, chapter 19. Chapman & Hall/CRC, 2005.
6. Sunil Arya, David M. Mount, and Eunhui Park. Approximate geometric MST range queries. In *Proceedings of the 31st International Symposium on Computational Geometry (SoCG 2015)*, pages 781–795, 2015.
7. Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k-median and facility location problems. *SIAM Journal on Computing*, 33(3):544–562, 2004.
8. Peter Brass, Christian Knauer, Chan-Su Shin, Michiel Smid, and Ivo Vigan. Range-aggregate queries for geometric extent problems. In *Proceedings of the 19th Computing: The Australasian Theory Symposium (CATS 2013)*, volume 141, pages 3–10, 2013.
9. Ke Chen. On coresets for k-median and k-means clustering in metric and euclidean spaces and their applications. *SIAM Journal on Computing*, 39(3):923–947, 2009.
10. Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag TELOS, 2008.

11 James R. Driscoll, Neil Sarnak, Daniel D. Sleator, and Robert E. Tarjan. Making data structures persistent. *Journal of Computer and System Sciences*, 38(1):86–124, 1989.

12 Tomás Feder and Daniel Greene. Optimal algorithms for approximate clustering. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC 1988)*, pages 434–444, 1988.

13 Dan Feldman and Michael Langberg. A unified framework for approximating and clustering data. In *Proceedings of the 43th Annual ACM Symposium on Theory of Computing (STOC 2011)*, pages 569–578, 2011.

14 Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38(Supplement C):293–306, 1985.

15 Prosenjit Gupta, Ravi Janardan, Yokesh Kumar, and Michiel Smid. Data structures for range-aggregate extent queries. *Computational Geometry*, 47(2, Part C):329–347, 2014.

16 Sariel Har-Peled. *Geometric Approximation Algorithms*. Mathematical surveys and monographs. American Mathematical Society, 2011.

17 Sariel Har-Peled and Akash Kushal. Smaller coresets for k-median and k-means clustering. *Discrete & Computational Geometry*, 37(1):3–19, Jan 2007.

18 Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC 2004)*, pages 291–300, 2004.

19 Anil Kumar Jain, M. Narasimha Murty, and Patrick J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

20 Jiří Matoušek. Geometric range searching. *ACM Computing Surveys*, 26(4):422–461, 1994.

21 Kurt Mehlhorn and Stefan Näher. Dynamic fractional cascading. *Algorithmica*, 5(1):215–241, 1990.

22 Yakov Nekrich and Michiel H. M. Smid. Approximating range-aggregate queries using coresets. In *Proceedings of the 22nd Annual Canadian Conference on Computational Geometry (CCCG 2010)*, pages 253–256, 2010.

23 Dimitris Papadias, Yufei Tao, Kyriakos Mouratidis, and Chun Kit Hui. Aggregate nearest neighbor queries in spatial databases. *ACM Transactions on Database System*, 30(2):529–576, 2005.

24 Jing Shan, Donghui Zhang, and Betty Salzberg. On spatial-range closest-pair query. In *Proceedings of the 8th International Symposium on Advances in Spatial and Temporal Databases (SSTD 2003)*, pages 252–269, 2003.

Point Location in Dynamic Planar Subdivisions

Eunjin Oh
Max Planck Institute for Informatics
Saarbrücken, Germany
eoh@mpi-inf.mpg.de
https://orcid.org/0000-0003-0798-2580

Hee-Kap Ahn
POSTECH
Pohang, Korea
heekap@postech.ac.kr
https://orcid.org/0000-0001-7177-1679

Abstract

We study the point location problem on dynamic planar subdivisions that allows insertions and deletions of edges. In our problem, the underlying graph of a subdivision is not necessarily connected. We present a data structure of linear size for such a dynamic planar subdivision that supports sublinear-time update and polylogarithmic-time query. Precisely, the amortized update time is $O(\sqrt{n}\log n(\log\log n)^{3/2})$ and the query time is $O(\log n(\log\log n)^2)$, where n is the number of edges in the subdivision. This answers a question posed by Snoeyink in the Handbook of Computational Geometry. When only deletions of edges are allowed, the update time and query time are just $O(\alpha(n))$ and $O(\log n)$, respectively.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases dynamic point location, general subdivision

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.63

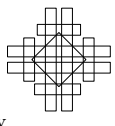
Related Version A full version of this paper is available at `https://arxiv.org/abs/1803.04325`.

Funding This research was supported by NRF grant 2011-0030044 (SRC-GAIA) funded by the government of Korea, and the MSIT (Ministry of Science and ICT), Korea, under the SW Starlab support program (IITP-2017-0-00905) supervised by the IITP (Institute for Information & communications Technology Promotion).

1 Introduction

Given a planar subdivision, a point location query asks with a query point specified by its coordinates to find the face of the subdivision containing the query point. In many situations such point location queries are made frequently, and therefore it is desirable to preprocess the subdivision and to store it in a data structure that supports point location queries fast.

The planar subdivisions for point location queries are usually induced by planar embeddings of graphs. A planar subdivision is connected if the underlying graph is connected. The vertices and edges of the subdivision are the embeddings of the nodes and arcs of the underlying graph, respectively. An edge of the subdivision is considered to be open, that is, it does not include its endpoints (vertices). A face of the subdivision is a maximal connected subset of the plane that does not contain any point on an edge or a vertex.




We say a planar subdivision *dynamic* if the subdivision allows two types of operations, the insertion of an edge to the subdivision and the deletion of an edge from the subdivision. The subdivision changes over insertions and deletions of edges accordingly. For an insertion of an edge e, we require e to intersect no edge or vertex in the subdivision and the endpoints of e to lie on no edge in the subdivision. We insert the endpoints of e in the subdivision as vertices if they were not vertices of the subdivision. In fact, the insertion with this restriction is general enough. The insertion of an edge e with an endpoint u lying on an edge e' of the subdivision can be done by a sequence of four operations: deletion of e', insertion of e, and insertions of two subedges of e' partitioned by u.

The dynamic point location problem is closely related to the dynamic vertical ray shooting problem [6]. For this problem, we are asked to find the edge of a dynamic planar subdivision that lies immediately above (or below) a query point. In the case that the subdivision is connected at any time, we can answer a point location query without increasing the space and time complexities using a data structure for the dynamic vertical ray shooting problem by maintaining the list of the edges incident to each face in a concatenable queue [6].

However, it is not the case in a general (possibly disconnected) planar subdivision. Although the dynamic vertical ray shooting algorithms presented in [2, 3, 5, 6] work for general (possibly disconnected) subdivisions, it is unclear how one can use them to support point location queries efficiently. As pointed out in some previous works [5, 6], a main issue concerns how to test whether the two edges lying immediately above two query points belong to the boundary of the same face in a dynamic planar subdivision. Notice that the boundary of a face may consist of more than one connected component.

In this paper, we consider a point location query on dynamic planar subdivisions. The subdivisions we consider are not necessarily connected, that is, the underlying graphs may consist of one or more connected components. We also require that every edge is a straight line segment. We present a data structure for a dynamic planar subdivision which answers point location queries efficiently.

Previous work. The dynamic vertical ray shooting problem has been studied extensively [2, 3, 5, 6]. These data structures do not require that the subdivision is connected, but they require that the subdivision is planar. None of the known algorithms for this problem is superior to the others. Moreover, optimal update and query times (or their optimal trade-offs) are not known. The update time or the query time (or both) is worse than $O(\log^2 n)$, except the data structures by Arge et al. [2] and by Chan and Nekrich [5]. The data structure by Arge at al. [2] supports expected $O(\log n)$ query time and expected $O(\log^2 n/\log\log n)$ update time under Las Vegas randomization in the RAM model. The data structure by Chan and Nekrich [5] supports $O(\log n(\log\log n)^2)$ query time and $O(\log n \log\log n)$ update time in the pointer machine model. Their algorithm can also be modified to reduce the query time at the expense of increasing the update time. As pointed out by Cheng and Janardan [6], all these data structures [2, 3, 5, 6] can be used for answering point location queries if the underlying graph of the subdivision is connected without increasing any resource.

Little has been known for the dynamic point location in general planar subdivisions. In fact, no nontrivial data structure is known for this problem.[1] Cheng and Janardan asked whether such a data structure can be maintained for a general planar subdivision [6], but

[1] The paper [2] claims that their data structure supports a point location query for a general subdivision. They present a vertical ray shooting data structure and claim that this structure supports a point location query for a general subdivision using the paper [9]. However, the paper [9] mentions that it works only for a subdivision such that every face in the subdivision has a constant complexity. Therefore, the point location problem for a general subdivision is still open.

this question has not been resolved until now. Very recently, it was asked again by Chan and Nekrich [5] and by Snoeyink [12]. Specifically, Snoeyink asked whether it is possible to construct a dynamic data structure for a general (possibly disconnected) planar subdivision supporting *sublinear query time* of determining if two query points lie in the same face of the subdivision.

Our result. In this paper, we present a data structure and its update and query algorithms for the dynamic point location in general planar subdivisions under the pointer machine model. This is the first result supporting sublinear update and query times, and answers the question posed in [5, 6, 12]. Precisely, the amortized update time is $O(\sqrt{n}\log n(\log\log n)^{3/2})$ and the query time is $O(\log n(\log\log n)^2)$, where n is the number of edges in the current subdivision. When only deletions of edges are allowed, the update and query times are just $O(\alpha(n))$ and $O(\log n)$, respectively. Here, we assume that a deletion operation is given with the *pointer* to an edge to be deleted in the current edge set.

Our approach itself does not require that every edge in the subdivision is a line segment, and can handle arbitrary curves of constant description. However, the data structures for dynamic vertical ray shooting queries require that every edge is a straight line segment, which we use as a black box. Once we have a data structure for answering vertical ray shooting queries for general curves, we can also extend our results to general curves. For instance, the result by Chan and Nekrich [5] is directly extended to x-monotone curves, and so is ours.

One may wonder if the problem is *decomposable* in the sense that a query over $D_1 \cup D_2$ can be answered in constant time from the answers from D_1 and D_2 for any pair of disjoint data sets D_1 and D_2 [8]. If a problem is decomposable, we can obtain a dynamic data structure from a static data structure of this problem using the framework of Bentley and Saxe [4], or Overmars and Leeuwen [10]. However, the dynamic point location problem in a general planar subdivision is not decomposable. To see this, consider a subdivision D consisting of a square face and one unbounded face. Let D_1 be the subdivision consisting of three edges of the square face and D_2 be the subdivision consisting of the remaining edge of the square face. There is only one face in D_1 (and D_2). Any two points in the plane are contained in the same face in D_1 (and D_2). But it is not the case for D. Therefore, the answers from D_1 and D_2 do not help to answer point location queries on D.

Outline. Consider any two query points in the plane. Our goal is to check whether they are in the same face of the current subdivision. To do this, we use the data structures for answering dynamic vertical ray shooting queries [2, 3, 5, 6], and find the edges lying immediately above the two query points. Then we are to check whether the two edges are on the boundary of the same face. In general subdivisions, the boundary of each face may consist of more than one connected components. This makes $\Theta(n)$ changes to the boundaries of the faces in the worst case, where n is the number of edges in the current subdivision. Therefore, we cannot maintain the explicit description of the subdivision.

To resolve this problem, we consider two different subdivisions, $\mathsf{M_o}$ and $\mathsf{M_n}$, such that the current subdivision consists of the edges of $\mathsf{M_o}$ and $\mathsf{M_n}$, and construct data structures on the subdivisions, $\mathsf{D_o}$ and $\mathsf{D_n}$, respectively. Recall that the dynamic point location problem is not decomposable. Thus the two subdivisions must be defined carefully. We set each edge in the current subdivision to be one of the three states: old, communal, and new. Then let $\mathsf{M_o}$ be the subdivision induced by all old and communal edges, and $\mathsf{M_n}$ be the subdivision induced by all new and communal edges. Note that every communal edge belongs to both subdivisions.

The state of each edge is defined as follows. The data structures are reconstructed periodically. In specific, they are rebuilt after processing $f(n)$ updates since the latest reconstruction, where n is the number of edges in the current subdivision. Here, $f(n)$ is called a reconstruction period, which is set to $\sqrt{n}$ roughly. When an edge e is inserted, we find the face F in $\mathsf{M_o}$ intersecting e and set the old edges on the outer boundary of F to communal. If one endpoint of e lies on the outer boundary of F and the other lies on an inner boundary of F, we set the old edges on this inner boundary to communal. Also, we set e to new. When an edge e is deleted, we find the faces in $\mathsf{M_o}$ incident to e and set the old edges of the outer boundaries of the faces to communal.

We show that the current subdivision has the following property: no face in the current subdivision contains both new and old edges on its outer boundary. In other words, for every face in the current subdivision, either every edge is classified as new or communal, or every edge is classified as old or communal. Due to this property, for any two query points, they are in the same face in the current subdivision if and only if they are in the same face in both $\mathsf{M_o}$ and $\mathsf{M_n}$. Therefore, we can represent the name of a face in the current subdivision as a pair of faces, one in $\mathsf{M_o}$ and one in $\mathsf{M_n}$. To answer a point location query on the current subdivision, it suffices to find the faces containing the query point in $\mathsf{M_o}$ and in $\mathsf{M_n}$.

To answer point location queries on $\mathsf{M_o}$, we observe that no edge is inserted to $\mathsf{M_o}$ unless it is rebuilt. Therefore, it suffices to construct a semi-dynamic point location data structure on $\mathsf{M_o}$ supporting only deletion operations. If only deletion operations are allowed, two faces are merged into one face, but no new face appears. Using this property, we provide a data structure supporting $O(\alpha(n))$ update time and $O(\log n)$ query time.

To answer point location queries on $\mathsf{M_n}$, we make use of the following property: the boundary of each face of $\mathsf{M_n}$ consists of $O(f(n))$ connected components while the number of edges of $\mathsf{M_n}$ is $\Theta(n)$ in the worst case, where n is the number of all edges in the current subdivision. Due to this property, the amount of the change on the subdivision $\mathsf{M_n}$ is $O(f(n))$ at any time. Therefore, we can maintain the explicit description of $\mathsf{M_n}$. In specific, we maintain a data structure on $\mathsf{M_n}$ supporting point location queries, which is indeed a doubly connected linked list of $\mathsf{M_n}$.

Due to lack of space, some proofs and details are omitted. The missing proofs and missing details can be found in the full version of the paper.

2 Preliminaries

Consider a planar subdivision M that consists of n straight line segment edges. Since the subdivision is planar, there are $O(n)$ vertices and faces. One of the faces of M is unbounded and all other faces are bounded. Notice that the boundary of a face is not necessarily connected. For the definitions of the faces and their boundaries, refer to [7, Chapter 2].

We consider each edge of the subdivision as two directed *half-edges*. The two half-edges are oriented in opposite directions so that the face incident to a half-edge lies to the left of it. In this way, each half-edge is incident to exactly one face, and the orientation of each connected component of the boundary of F is defined consistently. We call a boundary component of F the *outer boundary* of F if it is traversed along its half-edges incident to F in counterclockwise order around F. Except for the unbounded face, every face has a unique outer boundary. We call each connected component other than the outer boundary an *inner boundary* of F. Consider the outer boundary γ of a face. Since γ is a noncrossing closed curve, it subdivides the plane into regions exactly one of which contains F. We say a face F *encloses* a set C in the plane if C is contained in the (open) region containing F

of the planar subdivision induced by the outer boundary of F. Note that if F encloses F', the outer boundary of F does not intersect the boundary of F'. For more details on planar subdivisions, refer to the computational geometry book [7].

Our results are under the pointer machine model, which is more restrictive than the random access model. Under the pointer machine model, a memory cell can be accessed only through a series of pointers while any memory cell can be accessed in constant time under the random access model. Most of the results in [2, 3, 5, 6] are under the pointer machine model, and the others are under the random access model.

Updates: insertion and deletion of edges. We allow two types of update operations: InsertEdge(e) and DeleteEdge(e). In the course of updates, we maintain a current edge set E, which is initially empty. InsertEdge(e) is given with an edge e such that no endpoints of e lies on an edge of the current subdivision. This operation adds e to E, and thus update the current subdivision accordingly. Recall that an edge of the subdivision is a line segment excluding its endpoints. If an endpoint of e does not lie on a vertex of the current subdivision, we also add the endpoint of e to the current subdivision as a vertex. DeleteEdge(e) is given with an edge e in the current subdivision. Specifically, it is given with a pointer to e in the set E. This operation removes e from E, and updates the subdivision accordingly. If an endpoint of e is not incident to any other edge of the subdivision, we also remove the vertex which is the endpoint of e from the subdivision.

Queries. Our goal is to process update operations on the data structure so that given a query point q the face of the current subdivision containing q can be computed from the data structure efficiently. Specifically, each face is assigned a distinct name in the subdivision, and given a query point the name of the face containing the point is to be reported. A query of this type is called a *location query*, denoted by locate(x) for a query point x in the plane.

2.1 Data structures

In this paper, we show how to process updates and queries efficiently by maintaining a few data structures for dynamic planar subdivisions. In specific, we use disjoint-set data structures and concatenable queues. Before we continue with algorithms for updates and queries, we provide brief descriptions on these structures in the following. Throughout this paper, we use $S(n), U(n)$ and $Q(n)$ to denote the size, the update and query time of the data structures we use for the dynamic vertical ray shooting in a general subdivision. Notice that $U(n) = \Omega(\log n)$, $U(n) = o(n)$, and $Q(n) = \Omega(\log n)$ for any nontirivial data structure for the dynamic vertical ray shooting problem under the pointer machine model. Also, $U(n)$ is increasing. Thus in the following, we assume that $U(n)$ and $Q(n)$ satisfy these properties.

A disjoint-set data structure keeps track of a set of elements partitioned into a number of disjoint subsets [13]. Each subset is represented by a rooted tree in this data structure. The data structure has size linear in the total number of elements, and can be used to check whether two elements are in the same partition and to merge two partitions into one. Both operations can be done in $O(\alpha(N))$ time, where N is the number of elements at the moment and $\alpha(\cdot)$ is the inverse Ackermann function.

A concatenable queue represents a sequence of elements, and allows four operations: insert an element, delete an element, split the sequence into two subsequences, and concatenate two concatenable queues into one. By implementing them with 2-3 trees [1], we can support each operation in $O(\log N)$ time, where N is the number of elements at the moment. We can search any element in the queue in $O(\log N)$ time.

3 Deletion-only point location

In this section, we present a semi-dynamic data structure for point location queries that allows only DeleteEdge operations. Initially, we are given a planar subdivision consisting of n edges. Then we are given update operations DeleteEdge(e) for edges e in the subdivision one by one, and process them accordingly. In the course of updates, we answer point location queries. We maintain static data structures on the initial subdivision and a disjoint-set data structure that changes dynamically as we process DeleteEdge operations.

Static data structures. We construct the static point location data structure on the initial subdivision of size $O(n)$ in $O(n \log n)$ time [11]. Due to this data structure, we can find the face in the initial subdivision containing a query point in $O(\log n)$ time. We assign a name to each face, for instance, the integers from 1 to m for m faces. Also, we compute the doubly connected edge list of the initial subdivision, and make each edge in the current edge set to point to its counterparts in the doubly connected edge list. These data structures are static, so they do not change in the course of updates.

History structure of faces over updates. Consider an edge e to be deleted from the subdivision. If e is incident to two distinct faces in the current subdivision, the faces are merged into one and the subdivision changes accordingly. To apply such a change and keep track of the face information of the subdivision, we use a disjoint-set data structure S on the names of the faces in the initial subdivision. Initially, each face forms a singleton subset in S. In the course of updates, subsets in S are merged. Two elements in S are in the same subset of S if and only if the two faces corresponding to the two elements are merged into one face.

Deletion. We are given DeleteEdge(e) for an edge e of the subdivision. Since only history structure changes dynamically, it suffices to update the history structure only. We first compute the two faces in the initial subdivision that are incident to e in $O(1)$ time by using the doubly connected edge list. Then we check if the faces belong to the same subset or not in S. If they belong to two different subsets, we merge the subsets into one in $O(\alpha(n))$ time. The label of the root node in the merged subset becomes the name of the merged face. If the faces belong to the same subset, e is incident to the same face F in the current subdivision, and therefore there is no change to the faces in the subdivision, except the removal of e from the boundary of F. Since we do not maintain the boundary information of faces, there is nothing to do with the removal and we do not do anything on S. Thus, there is a bijection between faces in the current subdivision and subsets in the disjoint-set data structure S. We say that the face corresponding to the root of a subset *represents* the subset.

Location queries. To answer locate(x) for a query point x in the plane, we find the face F in the initial subdivision in $O(\log n)$ time. Then we return the subset in the disjoint-set data structure S that contains F in $O(\alpha(n))$ time. Precisely, we return the root of the subset containing F whose label is the name of the face containing x in the current subdivision. The argument in this section implies the correctness of the query algorithm.

▶ **Theorem 1.** *Given a planar subdivision consisting of n edges, we can construct data structures of size $O(n)$ in $O(n \log n)$ time so that* locate(x) *can be answered in $O(\log n)$ time for any point x in the plane and the data structures can be updated in $O(\alpha(n))$ time for a deletion of an edge from the subdivision.*

4 Data structures for fully dynamic point location

In Section 4 and Section 5, we present a data structure and its corresponding update and query algorithms for the dynamic point location in fully dynamic planar subdivisions. Initially, the subdivision is the whole plane. While we process a mixed sequence of insertions and deletions of edges, we maintain two data structures, one containing *old and communal edges* and one containing *new and communal edges*. We consider each edge of the subdivision to have one of three states, "new", "communal", and "old". The first data structure, denoted by $\mathsf{D_o}$, is the point location data structure on old and communal edges that supports only DeleteEdge operations described in Section 3. The second data structure, denoted by $\mathsf{D_n}$, is a fully dynamic point location data structure on new and communal edges.

Three states: old, communal and new. We rebuild both data structures periodically. When they are rebuilt, every edge in the current subdivision is set to old. As we process updates, some of them are set to communal as follows. For InsertEdge(e), there is exactly one face F of $\mathsf{M_o}$ whose interior is intersected by e as e does not intersect any edges or vertices. We set all edges on the outer boundary of F to communal. If e connects the outer boundary of F with one inner boundary of F, we set all edges on the inner boundary to communal. As a result, the outer boundary edges in the faces incident to e in $\mathsf{M_o}$ are communal or new after e is inserted. For DeleteEdge(e), there are at most two faces of $\mathsf{M_o}$ whose boundaries contain e. We set all edges on the outer boundary of these faces to communal. Here, we do not maintain the explicit description (the doubly connected edge list) of $\mathsf{M_o}$, but maintain the semi-dynamic data structure on $\mathsf{M_o}$ described in Section 3.

Also, the edges inserted after the latest reconstruction are set to new. The subdivision $\mathsf{M_n}$ of the new and communal edges has complexity of $\Theta(n)$ in the worst case. We maintain a fully dynamic point location data structure on $\mathsf{M_n}$, which is indeed the explicit description (the doubly connected edge list) of $\mathsf{M_n}$.

4.1 Reconstruction

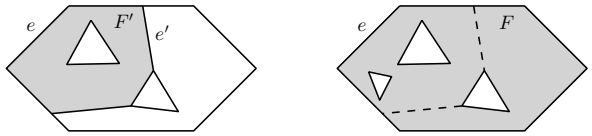
Let $f : \mathbb{N} \to \mathbb{N}$ be an increasing function satisfying that $f(n)/2 \leq f(n/2) \leq n/4$ for every n larger than a constant, which will be specified later. We call the function a *reconstruction period*. We reconstruct $\mathsf{D_o}$ and $\mathsf{D_n}$ if we have processed $f(n)$ updates since the latest reconstruction time, where n is the number of the edges in the current subdivision.

The following lemma is a key to achieve an efficient update time. Each face of $\mathsf{M_n}$ has $O(f(n))$ boundary components while the number of edge in $\mathsf{M_n}$ is $\Theta(n)$ in the worst case.

▶ **Lemma 2.** *Each face of* $\mathsf{M_n}$ *has* $O(f(n))$ *inner boundaries.*

Proof. Consider a face F of $\mathsf{M_n}$. There are two types of the inner boundaries of F: either all edges on an inner boundary are communal or at least one edge on an inner boundary is new. For an inner boundary of the second type, all edges other than the new edges are communal. By the construction, the number of new edges in $\mathsf{M_n}$ is at most $f(n)$. Recall that when we rebuild the data structures, $\mathsf{M_o}$ and $\mathsf{M_n}$, all edges are set to old.

We pick an arbitrary edge on each inner boundary of F of the first type, and call it the *representative* of the inner boundary. Each representative is inserted before the latest reconstruction, and is set to communal later. It becomes communal due to a pair (F', e'), where F' is a face of $\mathsf{M_o}$ and e' is an edge inserted or deleted after the latest reconstruction, such that the insertion or deletion of e' makes the edges on a boundary component of F' communal, and e was on this boundary component of F' at that moment. See Figure 1. If



(a) (b)

Figure 1 (a) Subdivision $\mathsf{M_o}$. All edges are old. (b) Subdivision $\mathsf{M_n}$. The two dashed edges are deleted after the reconstruction, which makes all outer boundary edges become communal. Then the edges on the leftmost triangle (hole) are inserted.

the representatives of all first-type inner boundaries of F are induced by distinct pairs, it is clear that the number of the first-type inner boundaries of F is $O(f(n))$. But it is possible that the representative of some inner boundaries of F are induced by the same pair (F', e').

Consider the representatives of some inner boundaries of F that are induced by the same pair (F', e'). The insertion of deletion of e' makes the outer boundary of F' become communal. In the case that e' connects the outer boundary of F' and an inner boundary of F', let γ be the cycle consisting of the outer boundary of F', the inner boundary of F' and e'. Let γ be the outer boundary of F', otherwise. All representatives induced by (F', e') are on γ, and therefore they are connected after e' is inserted or before e' is deleted. Notice that any two of such representatives are disconnected later. This means that γ becomes at least t connected components due to the removal of t edges on it, where t is the number of the representatives induced by (F', e'). The total number of edges that are removed after the latest reconstruction is $O(f(n))$, and each edge that are removed after the latest reconstruction can be a representative of at most two first-type inner boundaries of F. Therefore, the total number of the representatives of the first-type inner boundaries of F is also $O(f(n))$.

Consider an inner boundary of the second type. Since each edge is incident to at most one inner boundary of F, the number of the second-type inner boundaries is at most the number of new edges. Therefore, there are $O(f(n))$ second-type inner boundaries of $\mathsf{M_n}$. ◀

4.2 Two data structures

We maintain two data structures: $\mathsf{D_o}$, a semi-dynamic point location for old and communal edges, and $\mathsf{D_n}$, a fully dynamic point location for new and communal edges. In this subsection, we describe the data structures $\mathsf{D_o}$ and $\mathsf{D_n}$. The update procedures are described in Section 5.

Semi-dynamic point location for old and communal edges. After each reconstruction, we construct the point location data structure $\mathsf{D_o}$ supporting only DeleteEdge described in Section 3 for all edges in the current subdivision, which takes $O(n \log n)$ time. Recall that all edges in the current subdivision are old at this moment. In Section 5, we will see that the amortized time for reconstructing $\mathsf{D_o}$ is $O(n \log n / f(n))$ at any moment, where n is the number of all edges in the subdivision at the moment. As update operations are processed, some old or communal edges are deleted, and thus we remove them from $\mathsf{D_o}$. Notice that no edge is inserted to $\mathsf{D_o}$ by the definition of old and communal edges.

In addition to this, we store the old edges on each boundary component of the faces of $\mathsf{M_o}$ in a concatenable queue. Notice that such edges are not necessarily contiguous on the boundary component. In spite of this fact, we can traverse the old edges along a boundary component of each face of $\mathsf{M_o}$ in time linear in the number of the old edges due to the concatenable queue for the old edges.

Fully dynamic point location for new and communal edges. Let $\mathsf{E_n}$ be the set of all new and communal edges and $\mathsf{M_n}$ be the subdivision induced by $\mathsf{E_n}$. Also, let $\mathsf{E_o}$ denote the set of all old and communal edges and $\mathsf{M_o}$ be the subdivision induced by $\mathsf{E_o}$.

We maintain a dynamic data structure that supports vertical ray-shooting queries for $\mathsf{E_n}$. The update time $U(n)$ is $O(\log n \log\log n)$ and the query time $Q(n)$ is $O(\log n(\log\log n)^2)$ if we use the data structure by Chan and Nekrich [5]. Or, there are alternative data structures with different update and query times [2, 3, 6].

We also maintain the boundary of each face F of $\mathsf{M_n}$. We store each connected component of the boundary of F in a concatenable queue. More specifically, a concatenable queue represents a cyclic sequence of the edges in a connected component of the boundary of F. Since e is incident to at most two faces of $\mathsf{M_n}$, there are at most two such elements in the queues. We implement the concatenable queues using the 2-3 trees. We choose an element in each queue and call it the *root* of the queue. For a concatenable queue implemented by a 2-3 tree, we choose the root of the 2-3 tree as the root of the queue. Given any element of a queue, we can access the root of the queue in $O(\log n)$ time. For an inner boundary of a face F of $\mathsf{M_n}$, we let the root of the queue for this inner boundary point to the root of the queue for the outer boundary of F. We also make the root of the queue for the outer boundary of F point to the root of the queue for all inner boundaries of F. Also, we let each edge of $\mathsf{E_n}$ point to its corresponding elements in the queues.

We maintain a balanced binary search tree on the vertices of $\mathsf{M_n}$ sorted in a lexicographical order so that we can check whether a point in the plane is a vertex of $\mathsf{M_n}$ in $O(\log n)$ time. Also, for each vertex of $\mathsf{M_n}$, we maintain a balanced binary search tree on the edges incident to it in $\mathsf{M_n}$ in clockwise order around it. The update procedure of this data structure is straightforward, and the update time is subsumed by the time for maintaining the boundaries of the faces of $\mathsf{M_n}$. Thus, in the following, we do not mention the update of this structure.

▶ **Lemma 3.** *The data structures* $\mathsf{D_o}$ *and* $\mathsf{D_n}$ *have size* $O(n)$.

5 Update procedures for fully dynamic point location

We have two update operations: InsertEdge(e) and DeleteEdge(e). Recall that we rebuild the data structures periodically. More precisely, we reconstruct the data structures if we have processed $f(n)$ updates since the latest reconstruction time, where n is the number of the edges we have at the moment. After the reconstruction, the data structure $\mathsf{D_n}$ becomes empty. This is simply because the reconstruction resets all edges to old. For the data structure $\mathsf{D_o}$, we will show that the amortized time for reconstruction is $O(n \log n/f(n))$. Also, this data structure is updated as some old or communal edges are deleted.

In this section, we present a procedure for updates of the two data structures. Recall that we use $\mathsf{M_o}$ to denote the subdivision induced by the old and communal edges, and $\mathsf{M_n}$ to denote the subdivision induced by the new and communal edges. We use the subdivisions, $\mathsf{M_o}$ and $\mathsf{M_n}$, only for description purpose, and we do not maintain them.

5.1 Common procedure for edge insertions and edge deletions

We are given operation InsertEdge(e) or DeleteEdge(e) for an edge e. Recall that we construct $\mathsf{D_o}$ and $\mathsf{D_n}$ periodically. The reconstruction period $f : \mathbb{N} \to \mathbb{N}$ is an increasing function satisfying that $f(n)/2 \le f(n/2) \le n/4$ for every n larger than a constant.

▶ **Lemma 4.** *The amortized reconstruction time of* $\mathsf{D_o}$ *is* $O(n \log n/f(n))$*, where* n *is the number of all edges at the moment.*

The insertion or deletion sets some old edges to communal. By applying a point location query for an endpoint of e in $\mathsf{M_o}$, we find the faces F of $\mathsf{M_o}$ such that the boundary of F contains an endpoint of e or the interior of F is intersected by e. All edges lying on the outer boundary and at most one inner boundary of F become communal. We insert them and e to the data structure $\mathsf{D_n}$ for vertical ray shooting queries. This takes $O(N \cdot U(n))$ time, where N denotes the number of all edges inserted to the data structure. It is possible that some edges of the faces are already communal. In this case, we avoid removing (also accessing) such edges by using the concatenable queue representing the cyclic sequence of the old edges on each boundary component of F.

▶ **Lemma 5.** *The average number of old edges which are set to communal is $O(n/f(n))$ at any moment, where n is all edges at the moment.*

▶ **Corollary 6.** *The amortized time for inserting new and communal edges to the vertical ray shooting data structure in $\mathsf{D_n}$ is $O(n \cdot U(n)/f(n))$.*

5.2 Edge insertions

We are given operation InsertEdge(e) for an edge e. For the data structure $\mathsf{D_o}$, we do nothing since the set of the old and communal edges remains the same. For the data structure $\mathsf{D_n}$, we are required to insert one new edge e and several communal edges. In other words, we are required to update the ray shooting data structure, the concatenable queues and the pointers associated to each edge of $\mathsf{E_n}$. We first process the update due to the communal edges, and then process the update due to the new edge e. The process for the new edge e is the same as the process for the communal edges, except that there is only one new edge e, but there are $O(n/f(n))$ communal edges (amortized). In the following, we describe the process for the communal edges only.

Let $\overline{\mathsf{E}}_\mathsf{n}$ be the union of the closures of all edges of $\mathsf{E_n}$, where $\mathsf{E_n}$ is the set of the new and communal edges before InsertEdge(e) is processed. Recall that it is not necessarily connected. Recall that the old edges on the outer boundary of the face intersected by e become communal. If the outer boundary is connected to an inner boundary, we also set the edges of the inner boundary to communal. In this case, let γ be the cycle consisting of these two boundary components and e. Otherwise, let γ be the outer boundary of the face in $\mathsf{M_o}$ intersecting e. If γ consists of only communal edges, we do nothing. Thus we assume that it contains at least one old edge. We insert the old edges of γ to $\mathsf{D_n}$ in $O(n \cdot U(n)/f(n))$ amortized time by Corollary 6. Recall that the average number of such edges is $O(n/f(n))$ by Corollary 6.

Now we update the concatenable queues and the pointers made by the communal edges on γ in $O(f(n)Q(n) + n \log n/f(n))$. Let F be the face of $\mathsf{M_n}$ intersected by γ.

▶ **Lemma 7.** *The curve γ intersects no connected component of $\overline{\mathsf{E}}_\mathsf{n}$ enclosed by γ assuming that γ contains at least one old edge.*

By Lemma 7, there is a unique face F_γ in the subdivision $\mathsf{M_n}$ after the communal edges are inserted such that the outer boundary of F_γ is γ. See Figure 2. The boundaries of F change due to the communal edges, but the boundaries of the other faces remain the same. More precisely, F is subdivided into subfaces, one of which is F_γ. We compute the concatenable queues for each boundary component of the subfaces. We show how to do this for F_γ. While computing the boundary of F_γ, we can compute all boundaries of every subface in the same time. Details can be found in the full version of the paper.

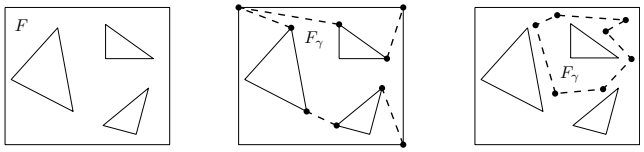


Figure 2 (a) Subdivision M_n before the communal edges are inserted. (b) The dashed edges are communal edges made by InsertEdge. They subdivide F into three faces one of which is F_γ. The boundary of F_γ is γ. (c) All edges of γ are communal edges set by InsertEdge.

Concatenable queue for the outer boundary γ of F_γ. We walk along the old edges of γ which become communal one by one using the concatenable queue for the old edges of γ. We make an empty concatenable queue for γ, and insert such edges one by one. If two consecutive old edges g_1 and g_2 of γ share no endpoints, there is a polygonal chain between g_1 and g_2 of γ consisting of communal edges only. Notice that this chain is a part of a boundary component of F. We find the boundary component of F in constant time. We split it with respect to g_1 and g_2, and combine one subchain with the concatenable queue for γ. We keep the other subchain for updating the boundary of F. In this way, we can obtain the concatenable queue for the outer boundary γ of F_γ. This takes $O(N \log n)$ time, where N is the number of old edges of γ which become communal.

Concatenable queues for the inner boundaries of F_γ. A inner boundary β of F might be enclosed by F_γ in the subdivision after the communal edges are inserted. For each inner boundary β of F, we check if it is enclosed by F_γ. To do this, we compute the edge e' immediately lying above the topmost vertex of β using the vertical ray shooting data structure on all new and communal edges, which include the edges of γ. Using the pointer for each edge e' pointing to the elements in the concatenable queues, we can find the boundary component $\beta_{e'}$ containing e' in constant time. If $\beta_{e'}$ is γ, we can determine if β is enclosed by F_γ immediately. Otherwise, β is enclosed by F_γ if and only if $\beta_{e'}$ is enclosed by F_γ. Therefore, we can determine for each inner boundary β of F whether β is enclosed by F_γ in $O(f(n)Q(n))$ time in total since there are $O(f(n))$ inner boundaries of F by Lemma 2.

Since each inner boundary of F_γ was an inner boundary of F before the communal edges are inserted, there is a concatenable queue for each inner boundary of F_γ whose root node points to the outer boundary of F. We make the root of the concatenable queue to point to F_γ, which takes $O(f(n))$ time in total for all inner boundaries of γ.

Pointers for edges. Finally, we update the pointers associated to each edge of E_n and each old edge of γ which become communal. Recall that each edge of E_n points to the element in the concatenable queues corresponding to it. For the update of the concatenable queues, we do not remove the elements of them. We just make their pointers to point to other elements of the queues. Therefore, we do not need to do anything for E_n. The only thing we do is to make each old edge of γ to point to the elements in the queues, one representing the outer boundary of F_γ and one representing a boundary component of a subface of F. This takes $O(N \log n)$ time, where N is the number of old edges of γ which become communal.

Therefore, the overall update time for inserting the communal edges is $O(f(n)Q(n) + N \log n)$. Since the average value of N is $O(n/f(n))$ by Lemma 5, the amortized update time is $O(f(n)Q(n) + n \log n/f(n))$. Similarly, the update time for the insertion of e is $O(f(n)Q(n) + \log n)$. The amortized reconstruction time is $O(n \log n/f(n))$ by Lemma 4.

Also, the amortized time for inserting the communal edges to the vertical ray shooting data structure is $O(n \cdot U(n)/f(n))$ by Corollary 6. Therefore, the overall update time is $O(f(n)Q(n) + n \log n/f(n) + n \cdot U(n)/f(n))$, which is $O(f(n)Q(n) + n \cdot U(n)/f(n))$ since $U(n) = \Omega(\log n)$.

▶ **Lemma 8.** *We can process* InsertEdge(e) *in* $O(f(n)Q(n) + n \cdot U(n)/f(n))$ *amortized time.*

5.3 Edge deletions

We are given operation DeleteEdge(e) for an edge e. For the data structure $\mathsf{D_o}$, we update the semi-dynamic point location data structure on the old and communal edges. We also update the concatenable queues for old edges on the boundary components of a face of $\mathsf{M_o}$.

For the data structure $\mathsf{D_n}$, we are required to update the concatenable queues and the pointers associated to each edge of $\mathsf{E_n}$. Here, we insert $O(n/f(n))$ communal edges and delete only one edge e from the data structure. The insertion of the communal edges is exactly the same as the case for edge insertions in the previous subsection. The deletion of e is also similar to the update procedure for edge insertions. The details can be found in the full version of the paper.

▶ **Lemma 9.** *We can process* DeleteEdge(e) *in* $O(f(n)Q(n) + n \cdot U(n)/f(n))$ *amortized time.*

6 Query procedure

We call the subdivision induced by all old, communal, and new edges the *complete subdivision* and denote it by $\mathsf{M_c}$. Sometimes we mention a face without specifying the subdivision if the face is in the complete subdivision.

Given $\mathsf{D_o}$ and $\mathsf{D_n}$, we are to answer Locate(x), that is, to find the face containing the query point x in $\mathsf{M_c}$. Let F_o and F_n be the faces of $\mathsf{M_o}$ and $\mathsf{M_n}$ containing x, respectively. By Theorem 1, we can find F_o in $O(\log n)$ time. For F_n, we find the edge e of $\mathsf{M_n}$ immediately lying above x in $Q(n)$ time using the vertical ray shooting data structure. Then we find the faces containing e on their boundaries using the pointers e has. There are at most two such faces. Since the connected components of the boundary of each face are oriented consistently, we can decide which one contains x in constant time. Therefore, we can compute F_n in $O(Q(n))$ time in total. To answer Locate(x), we need the following lemmas.

▶ **Lemma 10.** *No face of* $\mathsf{M_c}$ *contains both an old edge and new edge in its outer boundary.*

Proof. Assume to the contrary that both a new edge e_n and an old edge e_o lie on the outer boundary of a face F of $\mathsf{M_c}$. Note that e_n is inserted after the latest reconstruction. When e_n was inserted, all outer boundary edges of the face F' that was intersected by e_n in $\mathsf{M_o}$ at the moment were set to communal. Since a communal edge is set to old only by a reconstruction, the only possibility for e_o to remain as old is that e_o was not on the outer boundary of F' but on the outer boundary of another face in $\mathsf{M_o}$, and after then it has become an outer boundary edge of F by a series of splits and merges of the faces that are incident to e_o. These splits and merges occur only by insertions and deletions of edges, and e_o is set to communal by such a change to the face that is incident to e_o, and remains communal afterwards. This contradicts that e_o is old, and this case never occurs. ◀

▶ **Lemma 11.** *For any face F in $\mathsf{M_c}$, there exists a face in $\mathsf{M_o}$ or in $\mathsf{M_n}$ whose outer boundary coincides with the outer boundary of F.*

Using the two lemmas above, we can obtain the following lemma.

▶ **Lemma 12.** *For any two points in the plane, they are in the same face in* $\mathsf{M_c}$ *if and only if they are in the same face in* $\mathsf{M_o}$ *and in the same face in* $\mathsf{M_n}$*.*

Proof. Assume that two points x and y are in the same face F in $\mathsf{M_c}$. There is a face F' in $\mathsf{M_o}$ or in $\mathsf{M_n}$ whose outer boundary coincides with the outer boundary of F by Lemma 11. Consider a face F'' in $\mathsf{M_o}$ or $\mathsf{M_n}$ enclosed by F'. Neither x nor y is enclosed by F'' because the edge set of $\mathsf{M_o}$ (and $\mathsf{M_n}$) is a subset of the edge set of $\mathsf{M_c}$. Since the complete subdivision $\mathsf{M_c}$ is planar, this implies that x and y are in the same face in both $\mathsf{M_o}$ and $\mathsf{M_n}$.

Now assume that two points x and y are in different faces in $\mathsf{M_c}$. Let F_x and F_y be the faces containing x and y in $\mathsf{M_c}$, respectively. This means that x is not enclosed by F_y or y is not enclosed by F_x. The outer boundaries of F_x and F_y are distinct. By Lemma 11, there are faces F'_x and F'_y in $\mathsf{M_o}$ or in $\mathsf{M_n}$ whose outer boundaries coincide with the outer boundaries of F_x and F_y, respectively. Note that x is enclosed by F'_x and y is enclosed by F'_y. However, either x is not enclosed by F'_y or y is not enclosed by F'_x. Therefore, x and y are in different faces in $\mathsf{M_o}$ or $\mathsf{M_n}$, which proves the lemma. ◀

Lemma 12 immediately gives an $O(Q(n))$-time query algorithm. We represent the name of each face in $\mathsf{M_c}$ by the pair consisting of two faces, one from $\mathsf{M_o}$ and one from $\mathsf{M_n}$, corresponding to it. In this way, we have the following lemma.

▶ **Lemma 13.** *We can answer any query* LOCATE(x) *in* $O(Q(n))$ *time using* $\mathsf{D_o}$ *and* $\mathsf{D_n}$*.*

By setting $f(n) = \sqrt{n \cdot U(n)/Q(n)}$, we have the following theorem.

▶ **Theorem 14.** *We can construct a data structure of size* $O(S(n))$ *so that* LOCATE(x) *can be answered in* $O(Q(n))$ *time for any point* x *in the plane. Each update,* INSERTEDGE(e) *or* DELETEEDGE(e)*, can be processed in* $O(\sqrt{n \cdot U(n) \cdot Q(n)})$ *amortized time, where* n *is the number of edges at the moment.*

Using the data structure by Chan and Nekrich [5], we set $S(n) = n$, $Q(n) = \log n(\log\log n)^2$ and $U(n) = \log n \log\log n$.

▶ **Corollary 15.** *We can construct a data structure of size* $O(n)$ *so that* LOCATE(x) *can be answered in* $O(\log n(\log\log n)^2)$ *time for any point* x *in the plane. Each update,* INSERTEDGE(e) *or* DELETEEDGE(e)*, can be processed in* $O(\sqrt{n}\log n(\log\log n)^{3/2})$ *amortized time, where* n *is the number of edges at the moment.*

References

1. Alfred V. Aho and John E. Hopcroft. *The Design and Analysis of Computer Algorithms*. Addison-Wesley Longman Publishing Co., Inc., 1974.
2. Lars Arge, Gerth Stølting Brodal, and Loukas Georgiadis. Improved dynamic planar point location. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006)*, pages 305–314, 2006.
3. Hanna Baumgarten, Hermann Jung, and Kurt Mehlhorn. Dynamic point location in general subdivisions. *Journal of Algorithms*, 17(3):342–380, 1994.
4. Jon Louis Bentley and James B Saxe. Decomposable searching problems 1: Static-to-dynamic transformations. *Journal of Algorithms*, 1(4):301–358, 1980.
5. Timothy M. Chan and Yakov Nekrich. Towards an optimal method for dynamic planar point location. In *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2015)*, pages 390–409, 2015.

6 Siu-Wing Cheng and Ravi Janardan. New results on dynamic planar point location. *SIAM Journal on Computing*, 21(5):972–999, 1992.

7 Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag TELOS, 2008.

8 Jiří Matoušek. Efficient partition trees. *Discrete & Computational Geometry*, 8(3), 1992.

9 Mark H. Overmars. Range searching in a set of line segments. Technical report, Rijksuniversiteit Utrecht, 1983.

10 Mark H. Overmars and Jan van Leeuwen. Worst-case optimal insertion and deletion methods for decomposable searching problem. *Information Processing Letters*, 12(4):168–173, 1981.

11 Neil Sarnak and Robert E. Tarjan. Planar point location using persistent search trees. *Communications of the ACM*, 29(7):669–679, 1986.

12 Jack Snoeyink. Point location. In *Handbook of Discrete and Computational Geometry, Third Edition*, pages 1005–1023. Chapman and Hall/CRC, 2017.

13 Robert Endre Tarjan. Efficiency of a good but not linear set union algorithm. *Journal of the ACM*, 22(2):215–225, 1975.

Edge-Unfolding Nearly Flat Convex Caps

Joseph O'Rourke
Smith College, Department of Computer Science
Northampton, MA, USA
jorourke@smith.edu
https://orcid.org/0000-0001-5844-506X

Abstract

The main result of this paper is a proof that a nearly flat, acutely triangulated convex cap $\mathcal{C}$ in $\mathbb{R}^3$ has an edge-unfolding to a non-overlapping polygon in the plane. A *convex cap* is the intersection of the surface of a convex polyhedron and a halfspace. "Nearly flat" means that every outer face normal forms a sufficiently small angle $\phi < \Phi$ with the $\hat{z}$-axis orthogonal to the halfspace bounding plane. The size of Φ depends on the acuteness gap α: if every triangle angle is at most $\pi/2-\alpha$, then $\Phi \approx 0.36\sqrt{\alpha}$ suffices; e.g., for $\alpha = 3°$, $\Phi \approx 5°$. The proof employs the recent concepts of angle-monotone and radially monotone curves. The proof is constructive, leading to a polynomial-time algorithm for finding the edge-cuts, at worst $O(n^2)$; a version has been implemented.

2012 ACM Subject Classification Theory of computation → Computational geometry, Mathematics of computing → Graph theory

Keywords and phrases polyhedra, unfolding

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.64

Related Version A full version of this paper is available at [13], `http://arxiv.org/abs/1707.01006`.

Acknowledgements I benefited from discussions with Anna Lubiw and Mohammad Ghomi. I am grateful to four anonymous referees, who found an error in Lemma 5 and offered an alternative proof, shortened the justifications for Lemmas 3 and 6, suggested extensions and additional relevant references, and improved the exposition throughout.

1 Introduction

Let $\mathcal{P}$ be a convex polyhedron in $\mathbb{R}^3$, and let $\phi(f)$ be the angle the outer normal to face f makes with the $\hat{z}$-axis. Let H be a halfspace whose bounding plane is orthogonal to the $\hat{z}$-axis, and includes points vertically above that plane. Define a *convex cap* $\mathcal{C}$ of angle Φ to be $\mathcal{C} = \mathcal{P} \cap H$ for some $\mathcal{P}$ and H, such that $\phi(f) \le \Phi$ for all f in $\mathcal{C}$. We will only consider $\Phi < 90°$, which implies that the projection C of $\mathcal{C}$ onto the xy-plane is one-to-one. Note that $\mathcal{C}$ is not a closed polyhedron; it has no "bottom," but rather a boundary $\partial\mathcal{C}$.

Say that a convex cap $\mathcal{C}$ is *acutely triangulated* if every angle of every face is strictly acute, i.e., less than 90°. It may be best to imagine first constructing $\mathcal{P} \cap H$ and then acutely triangulating the surface. That every polyhedron may be acutely triangulated was first established by Burago and Zalgaller [4]. Recently Bishop proved that every PSLG (planar straight-line graph) of n vertices has a conforming acute triangulation, using $O(n^{2.5})$ triangles [2].[1] Applying Bishop's algorithm will create edges with flat (π) dihedral angles,

[1] His main Theorem 1.1 is stated for non-obtuse triangulations, but he says later that "the theorem also holds with an acute triangulation, at the cost of a larger constant in the $O(n^{2.5})$."

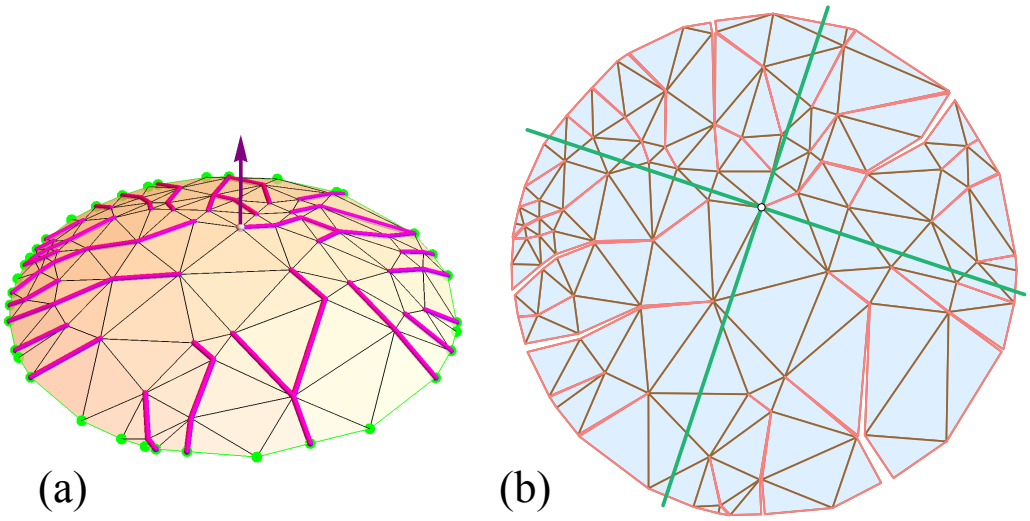


Figure 1 (a) A convex cap of 98 vertices, $\Phi \approx 33°$, with spanning forest $\mathcal{F}$ marked. $\mathcal{C}$ is non-obtusely triangulated (rather than acutely triangulated). (b) Edge-unfolding by cutting $\mathcal{F}$. The quadrant lines are explained in Section 5.2.

resulting from partitioning an obtuse triangle into several acute triangles. One might view the acuteness assumption as adding extra possible cut edges.

An *edge-unfolding* of a convex cap $\mathcal{C}$ is a cutting of edges of $\mathcal{C}$ that permits $\mathcal{C}$ to be developed to the plane as a simple (non-self-intersecting) polygon, a "net." The cut edges must form a boundary-rooted spanning forest $\mathcal{F}$: a forest of trees, each rooted on the boundary rim $\partial\mathcal{C}$, and spanning the internal vertices of $\mathcal{C}$. Our main result is:

▶ **Theorem 1.** *Every acutely triangulated convex cap $\mathcal{C}$ with face normals bounded by a sufficiently small angle Φ from the vertical, has an edge-unfolding to a non-overlapping polygon in the plane. The angle Φ is a function of the acuteness gap α (Eq. 6). The cut forest can be found in quadratic time.*

An example is shown in Fig. 1. Even if $\mathcal{C}$ is closed to a polyhedron by adding the convex polygonal base under $\mathcal{C}$, this polyhedron can be edge-unfolded without overlap [12].

1.1 Background

It is a long standing open problem whether or not every convex polyhedron has a non-overlapping edge-unfolding, often called Dürer's problem [6] [10]. Theorem 1 can be viewed as an advance on a narrow version of this problem. This theorem – without the acuteness assumption – has been a folk-conjecture for many years. A specific line of attack was conjectured in [9], and it is that sketch I follow for the proof here.

There have been two recent advances on Dürer's problem. The first is Ghomi's positive result that sufficiently thin polyhedra have edge-unfoldings [7]. This can be viewed as a counterpart to Theorem 1, which when supplemented by [12] shows that sufficiently flat polyhedra have edge-unfoldings. The second is a negative result that shows that when restricting cutting to geodesic "pseudo-edges" rather than edges of the polyhedral skeleton, there are examples that cannot avoid overlap [1].

It is natural to hope that Theorem 1 might lead to an edge-unfolding result for all acutely

triangulated convex polyhedra, but I have been so far unsuccessful in realizing this hope. Possible extensions are discussed in Section 11.

2 Overview of algorithm

We now sketch the simple algorithm in four steps; the proof of correctness will occupy the remainder of the paper. First, $\mathcal{C}$ is projected orthogonally to C in the xy-plane, with Φ small enough so that the acuteness gap of $\alpha > 0$ decreases to $\alpha' \leq \alpha$ but still $\alpha' > 0$. So C is acutely triangulated. Second, a boundary-rooted angle-monotone spanning forest F for C is found using the algorithm in [9]. Both the definition of angle-monotone and the algorithm will be described in Section 5 below, but for now we just note that each leaf-to-root path in F is both x- and y-monotone in a suitably rotated coordinate system. Third, F is lifted to a spanning forest $\mathcal{F}$ of $\mathcal{C}$, and the edges of $\mathcal{F}$ are cut. Finally, the cut $\mathcal{C}$ is developed flat in the plane. In summary: project, lift, develop.

I have not pushed on algorithmic time complexity, but certainly $O(n^2)$ suffices, as detailed in the full version [13].

3 Overview of proof

The proof relies on two results from earlier work: the angle-monotone spanning forest result in [9], and a radially monotone unfolding result in [11]. Those results are revised and explained as needed to allow this paper to stand alone. It is the use of angle-monotone and radially monotone curves and their properties that constitute the main novelties. The proof outline has these seven high-level steps, expanding upon the algorithm steps:

1. Project $\mathcal{C}$ to the plane containing its boundary rim, resulting in a triangulated convex region C. For sufficiently small Φ, C is again acutely triangulated.
2. Generalizing the result in [9], there is a θ-angle-monotone, boundary-rooted spanning forest F of C, for $\theta < 90°$. F lifts to a spanning forest $\mathcal{F}$ of the convex cap $\mathcal{C}$.
3. For sufficiently small Φ, both sides L and R of each cut-path $\mathcal{Q}$ of $\mathcal{F}$ are θ-angle-monotone when developed in the plane, for some $\theta < 90°$.
4. Any planar angle-monotone path for an angle $\leq 90°$, is radially monotone, a concept from [11].
5. Radial monotonicity of L and R, and sufficiently small Φ, imply that L and R do not cross in their planar development. This is a simplified version of a result from [11], and here extended to trees.
6. Extending the cap $\mathcal{C}$ to an unbounded polyhedron $\mathcal{C}^\infty$ ensures that the non-crossing of each L and R extends arbitrarily far in the planar development.
7. The development of $\mathcal{C}$ can be partitioned into θ-monotone "strips," whose side-to-side development layout guarantees non-overlap in the plane.

Through sometimes laborious arguments, I have tried to quantify steps even if they are in some sense obvious. Various quantities go to zero as $\Phi \to 0$. Those laborious arguments and other details are can be found in the full version [13].

3.1 Notation

I attempt to distinguish between objects in $\mathbb{R}^3$, and planar projected versions of those objects, either by using calligraphy ($\mathcal{C}$ in $\mathbb{R}^3$ vs. C in $\mathbb{R}^2$), or primes (γ in $\mathbb{R}^3$ vs. γ' in $\mathbb{R}^2$), and occasionally both ($\mathcal{Q}$ vs. Q'). Sometimes this seems infeasible, in which case we use different

symbols (u_i in $\mathbb{R}^3$ vs. v_i in $\mathbb{R}^2$). Sometimes we use $\perp$ as a subscript to indicate projections or developments of lifted quantities.

4 Projection angle distortion

1. Project $\mathcal{C}$ to the plane containing its boundary rim, resulting in a triangulated convex region C. For sufficiently small Φ, C is again acutely triangulated.

This first claim is obvious: Since every triangle angle is strictly less than 90°, and the distortion due to projection to a plane goes to zero as $\mathcal{C}$ becomes more flat, for some sufficiently small Φ, the acute triangles remain acute under projection.

In order to obtain a definite dependence on Φ, the following exact bound is derived in the full version [13].

▶ **Lemma 2.** *The maximum absolute value of the distortion $\Delta_\perp$ of any angle in $\mathbb{R}^3$ projected to the xy-plane, with respect to the tilt ϕ of the plane of that angle with respect to z, is given by:*

$$\Delta_\perp(\phi) \;=\; \cos^{-1}\left(\frac{\sin^2\phi}{\sin^2\phi - 2}\right) - \pi/2 \;\approx\; \phi^2/2 - \phi^4/12 + O(\phi^5)\,, \tag{1}$$

where the approximation holds for small ϕ.

In particular, $\Delta_\perp(\Phi) \to 0$ as $\Phi \to 0$. For example, $\Delta_\perp(10°) \approx 0.9°$.

5 Angle-monotone spanning forest

2. Generalizing the result in [9], there is a θ-angle-monotone, boundary rooted spanning forest F of C, for $\theta < 90°$. F lifts to a spanning forest $\mathcal{F}$ of the convex cap $\mathcal{C}$.

First we define angle-monotone paths, which originated in [5] and were further explored in [3], and then turn to the spanning forests we need here.

5.1 Angle-monotone paths

Let C be a planar, triangulated convex domain, with ∂C its boundary, a convex polygon. Let G be the (geometric) graph of all the triangulation edges in C and on ∂C.

Define the *θ-wedge* $W(\theta, v)$ to be the region of the plane bounded by rays separated by angular *width* θ emanating from v in fixed directions. W is closed along (i.e., includes) both rays. A polygonal path $Q = (v_0, \ldots, v_k)$ following edges of G is called *θ-angle-monotone* (or *θ-monotone* for short) if the vector of every edge (v_i, v_{i+1}) lies in $W(\theta, v_0)$ (and therefore $Q \subseteq W(\theta, v_0)$) in a fixed coordinate system.[2] If $\theta \leq 90°$, then a θ-monotone path is both x- and y-monotone in a suitable coordinate system, i.e., it meets every vertical, and every horizontal line in a point or a segment, or not at all.

[2] My notation here is slightly different from the notation in [9] and earlier papers, as I want to emphasize the reliance on θ.

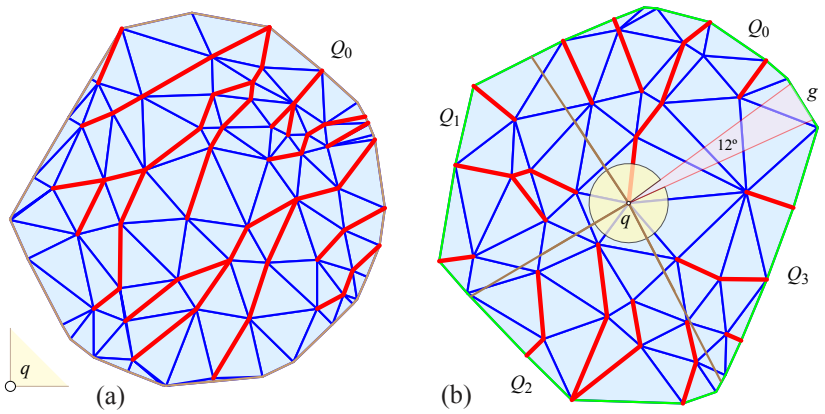


Figure 2 (a) q placed so that $C \subset Q_0$. (b) near-quadrants Q_i have width $\theta = 87°$, so the gap g has angle $4\alpha' = 12°$.

5.2 Angle-monotone spanning forest

It was proved in [9] that every non-obtuse triangulation G of a convex region C has a boundary-rooted spanning forest F of C, with all paths in F 90°-monotone. We describe the proof and simple construction algorithm before detailing the changes necessary for strictly acute triangulations.

Some internal vertex q of G is selected, and the plane partitioned into four 90°-quadrants Q_0, Q_1, Q_2, Q_3 by orthogonal lines through q. Each quadrant is closed along one axis and open on its counterclockwise axis; q is considered in Q_0 and not in the others, so the quadrants partition the plane. Then paths are grown within each quadrant independently, as follows. A path is grown from any vertex $v \in Q_i$ not yet included in the forest F_i, stopping when it reaches either a vertex already in F_i, or ∂C. These paths never leave Q_i, and result in a forest F_i spanning the vertices in Q_i . No cycle can occur because a path is grown from v only when v is not already in F_i; so v becomes a leaf of a tree in F_i. Then $F = F_1 \cup F_2 \cup F_3 \cup F_4$.

Because our acute triangulation is a non-obtuse triangulation, following the algorithm from [9] leads to angle-monotone paths for $\theta = 90° - \alpha' < 90°$. Although it is natural to place the quadrants origin q near the center of C, in fact choosing a q exterior to C so that all paths fall in the near-quadrant Q_0 suffices to determine F; see Fig. 2(a). The only reason to prefer a $q \in C$ is that this allows the conclusion mentioned earlier that closing $\mathcal{C}$ with a convex polygon base still permits an edge-unfolding of the closed polyhedron [12]. We leave the argument that shows q can be chosen at an interior vertex of C (see Fig. 2(b)) to the full version [13], and continue to illustrate $q \in C$.

We conclude this section with a lemma:

▶ **Lemma 3.** *If G is an acute triangulation of a convex region C, with acuteness gap α', then there exists a boundary-rooted spanning forest F of C, with all paths in F θ-angle-monotone, for $\theta = 90° - \alpha' < 90°$.*

6 Curve distortion

3. For sufficiently small Φ, both sides L and R of each cut-path $\mathcal{Q}$ of $\mathcal{F}$ are θ-angle-monotone when developed in the plane, for some $\theta < 90°$.

This step says, essentially, that each θ-monotone path Q' in the planar projection is not distorted much when lifted to $\mathcal{Q}$ on $\mathcal{C}$. This is obviously true as $\Phi \to 0$, but it requires proof. We need to establish that the left and right incident angles of the cut $\mathcal{Q}$ develop to the plane as still θ-monotone paths for some (larger) $\theta \le 90°$.

First we bound the total curvature of $\mathcal{C}$ to address the phrase, "For sufficiently small Φ, ..." The near flatness of the convex cap $\mathcal{C}$ is controlled by Φ, the maximum tilt of the normals from $\hat{z}$. Let ω_i be the curvature at internal vertex $u_i \in \mathcal{C}$ (i.e., 2π minus the sum of the incident angles to u_i), and $\Omega = \sum_i \omega_i$ the total curvature. We bound Ω as a function of Φ in the following lemma. (The reverse is not possible: even a small Ω could be realized with large Φ.)

▶ **Lemma 4.** *The total curvature $\Omega = \sum_i \omega_i$ of $\mathcal{C}$ satisfies*

$$\Omega \le 2\pi(1 - \cos\Phi) \approx \pi\Phi^2 - \pi\Phi^4/12 + O(\Phi^5) \ . \tag{2}$$

This is proved in the full version [13] as the area of a spherical cap on the Gaussian sphere for $\mathcal{C}$.

Our proof of limited curve lifting distortion uses the Gauss-Bonnet theorem,[3] in the form $\tau + \omega = 2\pi$: the turn of a closed curve plus the curvature enclosed is 2π.

To bound the curve distortion of Q', we need to bound the distortion of pieces of a closed curve that includes Q' as a subpath. Our argument here is not straightforward, but the conclusion is that, as $\Phi \to 0$, the distortion also $\to 0$:

▶ **Lemma 5.** *The difference in the total turn of any prefix of $\mathcal{Q}$ on the surface $\mathcal{C}$ from its planar projection Q' is bounded by $3\Delta_\perp + 2\Omega$ (Eq. 4), which, for small Φ, is a constant times Φ^2 (Eq. 5). Therefore, this turn goes to zero as $\Phi \to 0$.*

The reason the proof is not straightforward is that Q' could have an arbitrarily large number n of vertices, so bounding the angle distortion at each by $\Delta_\perp$ would lead to arbitrarily large distortion $n\Delta_\perp$. The same holds for the rim. So global arguments that do not cumulate errors seem necessary.

First we need a simple lemma, which is essentially the triangle inequality on the 2-sphere. Let $R' = \partial C$ and $R = \partial\mathcal{C}$ be the rims of the planar C and of the convex cap $\mathcal{C}$, respectively.

▶ **Lemma 6.** *The planar angle ψ' at a vertex v of the rim R' lifts to 3D angles of the triangles of the cap $\mathcal{C}$ incident to v, whose sum ψ satisfies $\psi \ge \psi'$.*

Now we use Lemma 6 to bound the total turn of the rim R of $\mathcal{C}$ and R' of C'. Although the rims are geometrically identical, their turns are not. The turn at vertex a' of the planar rim R' is $\pi - \psi'$, while the turn at vertex a of the 3D rim R is $\pi - \psi$. By Lemma 6, $\psi \ge \psi'$, so the turn at each vertex of the 3D rim R is at most the turn at each vertex of the 2D rim R'. Therefore the total turn of the 3D rim τ_R is smaller than or equal to the total turn of the 2D rim $\tau_{R'}$. And Gauss-Bonnet allows us to quantify this:

$$\tau_{R'} = 2\pi \ , \ \tau_R + \Omega = 2\pi \ , \ \tau_{R'} - \tau_R = \Omega \ .$$

For any subportion of the rims $r' \subset R'$, $r \subset R$, Ω serves as an upper bound, because we know the sign of the difference is the same at every vertex of r', r: $\tau_{r'} - \tau_r \le \Omega$.

[3] See, for example, Lee's description [8, Thm.9.3, p.164]. My τ is Lee's κ_N.

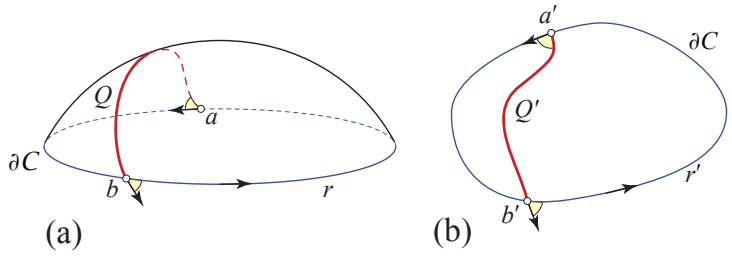


Figure 3 (a) C, the projection of the cap $\mathcal{C}$. (b) $\mathcal{Q}$ is the lift of Q' to $\mathcal{C}$.

6.1 Turn distortion of Q'

We need to bound $\Delta Q = |\tau'_Q - \tau_{\mathcal{Q}}|$, the turn difference between Q' in the plane and $\mathcal{Q}$ on the surface of $\mathcal{C}$, for Q' any prefix of an angle-monotone path in C that lifts to $\mathcal{Q}$ on $\mathcal{C}$. The reason for the prefix here is that we want to bound the turn of any segment of Q', not just the last segment, whose turn is $\sum_i \tau_i$. And note that there can be cancellations among the τ_i along Q', as we have no guarantee that they are all the same sign.

First we sketch the situation if Q cut all the way across $\mathcal{C}$, as illustrated in Fig. 3(a). We apply the Gauss-Bonnet theorem: $\tau + \omega = 2\pi$, where $\omega \leq \Omega$ is the total curvature inside the path $Q \cup r$, and then the planar projection (Fig. 3(b)), we have:

$$\begin{aligned} \tau + \omega &= \tau_Q + (\tau_a + \tau_b) + \tau_r + \omega = 2\pi \\ \tau + \omega &= \tau_{Q'} + (\tau_{a'} + \tau_{b'}) + \tau_{r'} + 0 = 2\pi \end{aligned} \tag{3}$$

Subtracting these equations will lead to a bound on ΔQ.

But, as indicated, Q does not cut all the way across $\mathcal{C}$, and we need to bound ΔQ for any prefix of Q (which we will still call Q). Let Q cut from $a \in \mathcal{C}$ to $b \in \partial\mathcal{C}$. We truncate $\mathcal{C}$ by intersecting with a halfspace whose bounding plane H includes a, as in Fig. 4(a). It is easy to arrange H so that $H \cap Q = \{a\}$, i.e., so that H does not otherwise cut Q, as follows. First, in projection, Q' falls inside $\overline{W}(\theta, a')$, the backward wedge passing through a'. Then start with H vertical and tangent to this wedge at a, and rotate it out to reaching $\partial\mathcal{C}$ as illustrated. The result is a truncated cap $\mathcal{C}_T$. We connect a to a point c on the new $\partial\mathcal{C}_T$, depicted abstractly in Fig. 4(b). Now we perform the analogous calculation for the curve $Q \cup r_1 \cup ca$ on $\mathcal{C}$, and $\mathcal{Q}' \cup r'_1 \cup ca'$:

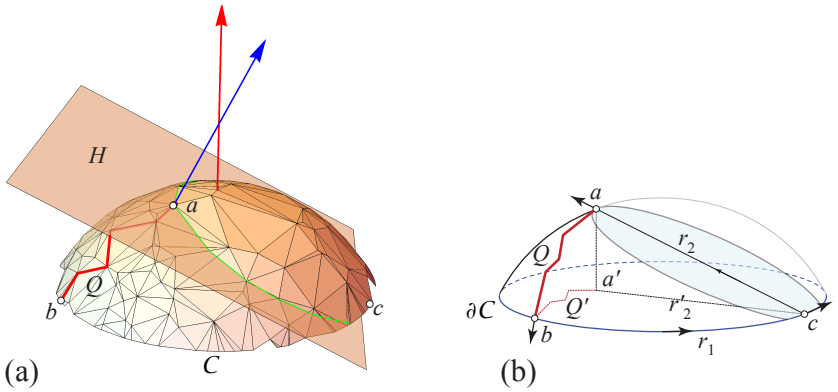


Figure 4 (a) Truncating $\mathcal{C}$ with H so that $H \cap \mathcal{Q} = \{a\}$. (b) $r_2 = ac$ and $r'_2 = a'c$.

$$\begin{aligned} \tau_{Q'} + (\tau_{a'} + \tau_{b'} + \tau_{c'}) + (\tau_{r'_1} + \tau_{r'_2}) + 0 &= 2\pi \\ \tau_{\mathcal{Q}} + (\tau_a + \tau_b + \tau_c) + (\tau_{r_1} + \tau_{r_2}) + \omega &= 2\pi \end{aligned}$$

Subtracting leads to

$$\begin{aligned} \tau_{Q'} - \tau_{\mathcal{Q}} &= ((\tau_a - \tau_{a'}) + (\tau_b - \tau_{b'}) + (\tau_c - \tau_{c'})) + (\tau_{r_1} - \tau_{r'_1}) + (\tau_{r_2} - \tau_{r'_2}) + \omega \\ \Delta Q &\leq 3\Delta_\perp + 2\Omega \end{aligned} \tag{4}$$

The logic of the bound is: (1) Each of the turn distortions at a, b, c is at most $\Delta_\perp$. (2) The r_1 turn difference is bounded by $\omega \leq \Omega$. And (3) $\tau_{r_2} = \tau_{r'_2} = 0$. Using the small-$\Phi$ bounds derived earlier in Eqs. 1 and 2:

$$|\Delta Q| \;\leq\; 3\Delta_\perp + 2\Omega \;\approx\; (2\pi + \tfrac{3}{2})\Phi^2 \,. \tag{5}$$

Thus we have $\Delta Q \to 0$ as $\Phi \to 0$, as claimed.

We finally return to the claim at the start of this section: For sufficiently small Φ, both sides L and R of each path $\mathcal{Q}$ of $\mathcal{F}$ are θ-angle-monotone when developed in the plane, for some $\theta < 90°$.

The turn at any vertex of $\mathcal{Q}$ is determined by the incident face angles to the left following the orientation shown in Fig. 3, or to the right reversing that orientation (clearly the curvature enclosed by either curve is $\leq \Omega$). These incident angles determine the left and right planar developments, L and R, of $\mathcal{Q}$. Because we know that Q' is θ-angle-monotone for $\theta < 90°$, there is some finite "slack" $\alpha = 90° - \theta$. Because Lemma 5 established a bound for any prefix of $\mathcal{Q}$, it bounds the turn distortion of each edge of $\mathcal{Q}$, which we can arrange to fit inside that slack. So the bound provided by Lemma 5 suffices to guarantee that:

▶ **Lemma 7.** *For sufficiently small Φ, both L and R remain θ-angle-monotone for some (larger) θ, but still $\theta \leq 90°$.*

To ensure $\theta \leq 90°$, we need that the maximum distortion fits into the acuteness gap: $|\Delta Q| \leq \alpha$. Using Eq. 5 leads to:

$$\Phi \;\leq\; \sqrt{\frac{2}{4\pi + 3}}\sqrt{\alpha} \;\approx\; 0.36\sqrt{\alpha} \,. \tag{6}$$

For example, if all triangles are acute by $\alpha = 4°$, then $\Phi \approx 5.4°$ suffices.

That F lifts to a spanning forest $\mathcal{F}$ of the convex cap $\mathcal{C}$ is immediate. What is not straightforward is establishing the requisite properties of $\mathcal{F}$.

7 Radially monotone paths

4. Any planar angle-monotone path for an angle $\leq 90°$, is radially monotone, a concept from [11].

Let $Q = (v_0, v_1, \ldots, v_k)$ be a simple (non-self-intersecting) directed path of edges of C connecting an interior vertex v_0 to a boundary vertex $v_k \in \partial C$. We say that Q is *radially monotone* with respect to (w.r.t.) v_0 if the distances from v_0 to all points of Q are (non-strictly) monotonically increasing. We define path Q to be *radially monotone* (without qualification) if it is radially monotone w.r.t. each of its vertices: $v_0, v_1, \ldots, v_{k-1}$. It is an

easy consequence of these definitions that, if Q is radially monotone, it is radially monotone w.r.t. any point p on Q, not only w.r.t. its vertices.

Before proceeding, we discuss its intuitive motivation. If a path Q is radially monotone, then "opening" the path with sufficiently small curvatures ω_i at each v_i will avoid overlap between the two halves of the cut path. Whereas if a path is not radially monotone, then there is some opening curvature assignments ω_i to the v_i that would cause overlap: assign a small positive curvature $\omega_j > 0$ to the first vertex v_j at which radial monotonicity is violated, and assign the other vertices zero or negligible curvatures. Thus radially monotone cut paths are locally (infinitesimally) opening "safe," and non- radially monotone paths are potentially overlapping.[4]

The condition for Q to be radially monotone w.r.t. v_0 can be interpreted as requiring Q to cross every circle centered on v_0 at most once; see Fig. 5. The concentric circles viewpoint makes it evident that infinitesimal rigid rotation of Q about v_0 to Q' ensures that $Q \cap Q' = \{v_0\}$, for each point of Q simply moves along its circle. Of course the concentric circles must be repeated, centered on every vertex v_i.

7.1 Angle-monotone chains are radially monotone

Fig. 5(c) illustrates why a θ-monotone chain Q, for any $\theta \leq 90°$, is radially monotone: the vector of each edge of the chain points external to the quarter-circle passing through each v_i. And so the chain intersects the v_0-centered circles at most once. Thus Q is radially monotone w.r.t. v_0. But then the argument can be repeated for each v_i, for the wedge $W(v_i)$ is just a translation of $W(\theta, v_0)$.

It should be clear that these angle-monotone chains are special cases of radially monotone chains. But we rely on the spanning-forest theorem in [9] to yield angle-monotone chains, and we rely on the unfolding properties of radially monotone chains from [11] to establish non-overlap. We summarize in a lemma:

[4] The phrase "radial monotonicity" has also appeared in the literature meaning radially monotone w.r.t. just v_0, most recently in [7]. The version here is more stringent to guarantee non-overlap.

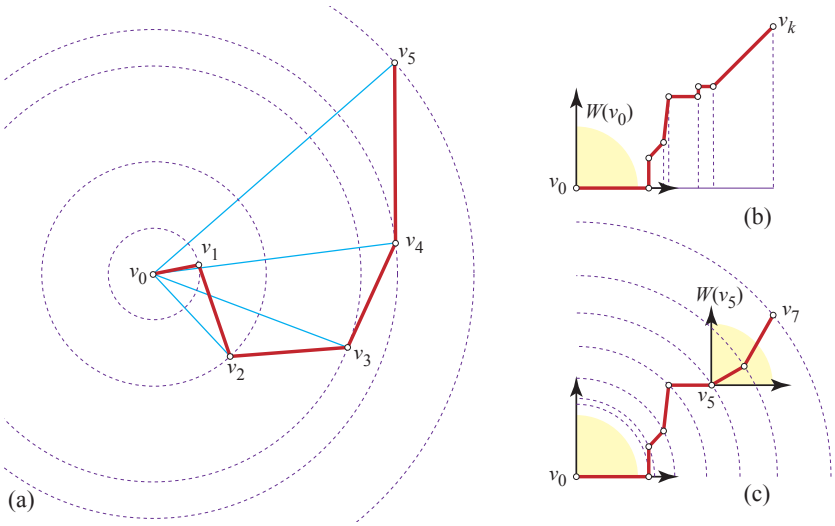


Figure 5 (a) A radially monotone chain, with its monotonicity w.r.t. v_0 illustrated. (b) A 90°-monotone chain, with x-monotonicity indicated. (c) Such a chain is also radially monotone.

▶ **Lemma 8.** *A θ-monotone chain Q, for any $\theta \le 90°$, is radially monotone.*

8 Noncrossing L & R developments

> 5. Radial monotonicity of L and R, and sufficiently small Φ, imply that L and R do not cross in their planar development. This is a simplified version of a result from [11], and here extended to trees.

We will use $\mathcal{Q} = (u_0, u_1, \ldots, u_k)$ as a path of edges on $\mathcal{C}$, with each $u_i \in \mathbb{R}^3$ a vertex and each u_iu_{i+1} an edge of $\mathcal{C}$. Let $Q = (v_0, v_1, \ldots, v_k)$ be a chain in the plane. Define the *turn angle* τ_i at v_i to be the counterclockwise angle from $v_i - v_{i-1}$ to $v_{i+1} - v_i$. Thus $\tau_i = 0$ means that v_{i-1}, v_i, v_{i+1} are collinear. $\tau_i \in (-\pi, \pi)$; simplicity excludes $\tau_i = \pm\pi$.

Each turn of the chain Q sweeps out a sector of angles. We call the union of all these sectors $\Lambda(Q)$; this forms a cone such that, when apexed at v_0, $Q \subseteq \Lambda(Q)$. The rays bounding $\Lambda(Q)$ are determined by the segments of Q at extreme angles; call these angles $\sigma_{\max}$ and $\sigma_{\min}$. See ahead to Fig. 6(a) for an example. Let $|\Lambda(Q)|$ be the measure of the apex angle of the cone, $\sigma_{\max} - \sigma_{\min}$. We will assume that $|\Lambda(Q)| < \pi$ for our chains Q, although it is quite possible for radially monotone chains to have $|\Lambda(Q)| > \pi$. In our case, in fact $|\Lambda(Q)| < \pi/2$, but that tighter inequality is not needed for Theorem 9 below. The assumption $|\Lambda(Q)| < \pi$ guarantees that Q fits in a halfplane H_Q whose bounding line passes through v_0.

Because $\sigma_{\min}$ is turned to $\sigma_{\max}$, we have that the total absolute turn $\sum_i |\tau_i| \ge |\Lambda(Q)|$. But note that the sum of the turn angles $\sum_i \tau_i$ could be smaller than $|\Lambda(Q)|$ because of cancellations.

8.1 The left and right planar chains L & R

Let ω_i be the curvature at vertex u_i of $\mathcal{Q}$. We view u_0 as a leaf of a cut forest, which will then serve as the end of a cut path, and the "source" of opening that path.

Let λ_i be the surface angle at u_i left of $\mathcal{Q}$, and ρ_i the surface angle right of $\mathcal{Q}$ there. So $\lambda_i + \omega_i + \rho_i = 2\pi$, and $\omega_i \ge 0$. Define L to be the planar path from the origin with left angles λ_i, R the path with right angles ρ_i. These paths are the left and right planar developments of $\mathcal{Q}$. We label the vertices of the developed paths ℓ_i, r_i.

Define $\omega(\mathcal{Q}) = \sum_i \omega_i$, the total curvature along the path $\mathcal{Q}$. We will assume $\omega(\mathcal{Q}) < \pi$, a very loose constraint in our nearly flat circumstances. For example, with $\Phi = 30°$, Ω for $\mathcal{C}$ is $< \pi\Phi^2 \approx 49°$, and $\omega(\mathcal{Q})$ can be at most Ω.

8.2 Left-of definition

Let $A = (a_0, \ldots, a_k)$ and $B = (b_0, \ldots, b_k)$ be two (planar) radially monotone chains sharing $x = a_0 = b_0$. (Below, A and B will be the L and R chains.) Let $D(r)$ be the circle of radius r centered on x. $D(r)$ intersects any radially monotone chain in at most one point. Let a and b be two points on $D(r)$. Say that a is *left of* b, $a \preceq b$, if the counterclockwise arc from b to a is less than π. If $a = b$, then $a \preceq b$. Now we extend this relation to entire chains. Say that chain A is *left of* B, $A \preceq B$, if, for all $r > 0$, if $D(r)$ meets both A and B, in points a and b respectively, then $a \preceq b$. If $D(r)$ meets neither chain, or only one, no constraint is specified. Note that, if $A \preceq B$, A and B can touch but not properly cross.

8.3 Noncrossing theorem

▶ **Theorem 9.** *Let $\mathcal{Q}$ be an edge cut-path on $\mathcal{C}$, and L and R the developed planar chains derived from $\mathcal{Q}$, as described above. Under the assumptions:*

1. *Both L and R are radially monotone,*
2. *The total curvature along $\mathcal{Q}$ satisfies $\omega(\mathcal{Q}) < \pi$.*
3. *Both cone measures are less than π: $|\Lambda(L)| < \pi$ and $|\Lambda(R)| < \pi$,*

then $L \preceq R$: L and R may touch and share an initial chain from $\ell_0 = r_0$, but L and R do not properly cross, in either direction.

That the angle conditions (2) and (3) are necessary is shown in the full version [13].

Proof. We first argue that L cannot wrap around and cross R from its right side to its left side. (Illustrations supporting this proof are in the full version [13].) Let $\rho_{\max}$ be the counterclockwise bounding ray of $\Lambda(R)$. In order for L to enter the halfplane H_R containing $\Lambda(R)$, and intersect R from its right side, $\rho_{\max}$ must turn to be oriented to enter H_R, a turn of $\geq \pi$. We can think of the effect of ω_i as augmenting R's turn angles τ_i to L's turn angles $\tau'_i = \tau_i + \omega_i$. Because $\omega_i \geq 0$ and $\omega(\mathcal{Q}) = \sum_i \omega_i < \pi$, the additional turn of the chain segments of R is $< \pi$, which is insufficient to rotate $\rho_{\max}$ to aim into H_R. (Later (Section 9) we will see that we can assume L and R are arbitrarily long, so there is no possibility of L wrapping around the end of R and crossing R right-to-left.)

Next we show that L cannot cross R from left to right. We imagine Q right-developed in the plane, so that $Q = R$. We then view L as constructed from a fixed R by successively opening/turning the links of R by ω_i counterclockwise about r_i, with i running backwards from r_{n-1} to r_0, the source vertex of R. Fig. 6(b) illustrates this process. Let $L_i = (\ell_i, \ell_{i+1}, \ldots, \ell_k)$ be the resulting subchain of L after rotations $\omega_{n-1}, \ldots, \omega_i$, and R_i the corresponding subchain of $R = (r_i, r_{i+1}, \ldots, r_k)$, with $\ell_i = r_i$ the common source vertex. We prove $L_i \preceq R_i$ by induction.

$L_{n-1} \preceq R_{n-1}$ is immediate because $\omega_{n-1} \leq \omega(\mathcal{Q}) < \pi$. Assume now $L_{i+1} \preceq R_{i+1}$, and consider L_i. Because both L_i and R_i are radially monotone, circles centered on $\ell_i = r_i$ intersect the chains in at most one point each. L_i is constructed by rotating L_{i+1} rigidly by ω_i counterclockwise about $\ell_i = \rho_i$; see Fig. 6(b). This only increases the arc distance between the intersections with those circles, because the circles must pass through the gap representing $L_{i+1} \preceq R_{i+1}$, shaded in Fig. 6(a). And because we already established that L cannot enter the R halfplane H_R, we know these arcs are $< \pi$: for an arc of $\geq \pi$ could turn $\rho_{\max}$ to aim into H_R. So $L_i \preceq R_i$. Repeating this argument back to $i = 0$ yields $L \preceq R$, establishing the theorem. ◀

Our cut paths are (in general) leaf-to-root paths in some tree $\mathcal{T} \subseteq \mathcal{F}$ of the forest, so we need to extend Theorem 9 to trees.[5] The proof of the following is in the full version [13].

▶ **Corollary 10.** *The $L \preceq R$ conclusion of Theorem 9 holds for all the paths in a tree $\mathcal{T}$: $L' \preceq R$, for any such L'. (See Fig. 6(c,d).)*

[5] This extension was not described explicitly in [11].

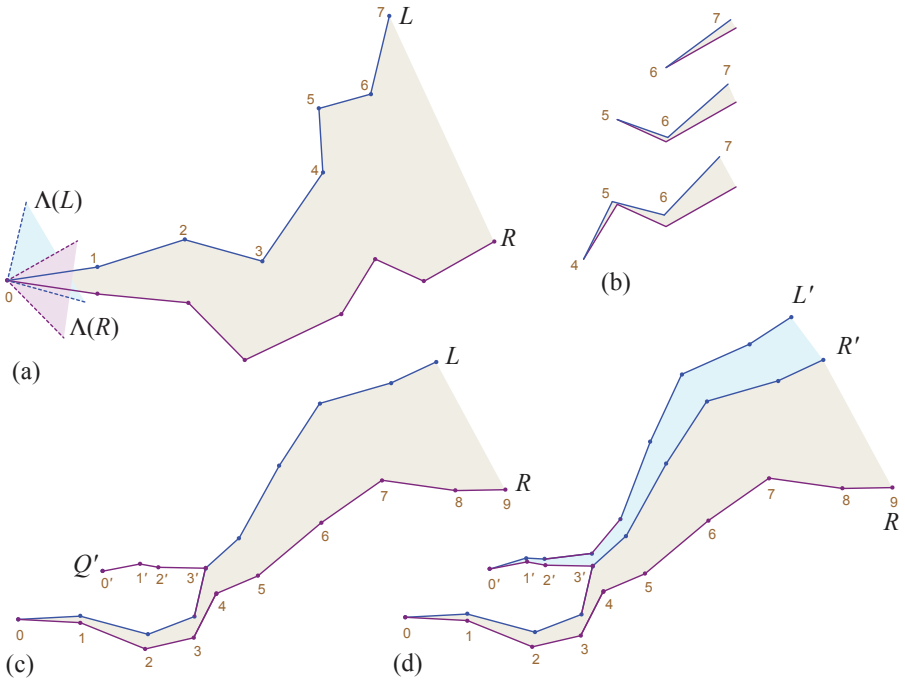


Figure 6 (a) Example 1: After opening Q to L and R. (b) Example 1: First steps in the induction proof. (c) Example 2: Q' joins Q at $v_3' = v_4$. After opening Q to L and R. (d) Example 2: After opening Q'.

9 Extending $\mathcal{C}$ to $\mathcal{C}^\infty$

6. Extending the cap $\mathcal{C}$ to an unbounded polyhedron $\mathcal{C}^\infty$ ensures that the non-crossing of each L and R extends arbitrarily far in the planar development.

In order to establish non-overlap of the unfolding, it will help to extend the convex cap $\mathcal{C}$ to an unbounded polyhedron $\mathcal{C}^\infty$ by extending the faces incident to the boundary $\partial\mathcal{C}$. The details are in the full version [13]. The consequence is that each cut path $\mathcal{Q}$ can be viewed as extending arbitrarily far from its source on $\mathcal{C}$. This technical trick permits us to ignore "end effects" as the cuts are developed in the next section.

10 Angle-monotone strips partition

7. The development of $\mathcal{C}$ can be partitioned into θ-monotone "strips," whose side-to-side development layout guarantees non-overlap in the plane.

The final step of the proof is to partition the planar C (and so the cap $\mathcal{C}$ by lifting) into strips that can be developed side-by-side to avoid overlap. We return to the spanning forest F of C (graph G), as discussed in Section 5.2. Define an *angle-monotone strip* (or more specifically, a θ-monotone strip) S as a region of C bound by two angle-monotone paths L_S and R_S which emanate from the quadrant origin vertex $q \in L_S \cap R_S$, and whose interior is vertex-free. The strips we use connect from q to each leaf $\ell \in F$, and then follow to the tree's root on ∂C. A simple algorithm to find such strips is described in the full version [13].

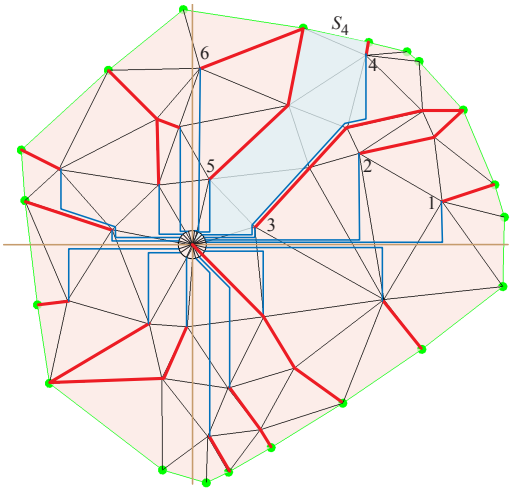


Figure 7 Waterfall strips partition. The S_4 strip highlighted.

see Fig. 7. Extending the $\preceq$ relation (Section 8.2) from curves $L \preceq R$ to adjacent strips, $S_i \preceq S_{i-1}$, shows that side-by-side layout of these strips develops all of $\mathcal{C}$ without overlap. This finally proves Theorem 1.

11 Discussion

It is natural to hope that Theorem 1 can be strengthened. That the rim of $\mathcal{C}$ lies in a plane is unlikely to be necessary: I believe the proof holds as long as shortest paths from q reach every point of $\partial\mathcal{C}$. Although the proof requires "sufficiently small Φ," limited empirical exploration suggests Φ need not be that small. (The proof assumes the worst case, with all curvature concentrated on a single path.) The assumption that $\mathcal{C}$ is acutely triangulated seems overly cautious. It seems feasible to circumvent the somewhat unnatural projection/lift steps with direct reasoning on the surface $\mathcal{C}$.

It is natural to wonder[6] if Theorem 1 leads to some type of "fewest nets" result for a convex polyhedron $\mathcal{P}$ [6, OpenProb.22.2, p.309]. At this writing I have a proof outline that, if successful, leads to the following (weak) result: If the maximum angular separation between face normals incident to any vertex leads to $\phi_{\max}$, and if the acuteness gap α accommodates $\phi_{\max}$ according to Eq. 6, then $\mathcal{P}$ may be unfolded to $\lesssim 1/\phi_{\max}^2$ non-overlapping nets. For example, $n = 2000$ random points on a sphere leads to $\phi_{\max} \approx 7.1°$ and if $\alpha \geq 6.9°$ – i.e., $\theta \leq 83.1°$ – then 64 non-overlapping nets suffice to unfold $\mathcal{P}$. The novelty here is that this is independent of the number of vertices n. The previous best result is $\lceil \frac{4}{11} F \rceil = \Omega(n)$ nets [14], where F is the number of faces of $\mathcal{P}$, which in this example leads to 1454 nets. However, the assumption that the acuteness gap α accommodates $\phi_{\max}$ restricts the applicability of this conjectured result.

References

1 Nicholas Barvinok and Mohammad Ghomi. Pseudo-edge unfoldings of convex polyhedra. arXiv:1709.04944: `https://arxiv.org/abs/1709.04944`, 2017.

2 Christopher J. Bishop. Nonobtuse triangulations of PSLGs. *Discrete & Comput. Geom.*, 56(1):43–92, 2016.

[6] Stefan Langerman, personal communication, August 2017.

3 Nicolas Bonichon, Prosenjit Bose, Paz Carmi, Irina Kostitsyna, Anna Lubiw, and Sander Verdonschot. Gabriel triangulations and angle-monotone graphs: Local routing and recognition. In *Internat. Symp. Graph Drawing Network Vis.*, pages 519–531. Springer, 2016.

4 Ju. D. Burago and V. A. Zalgaller. Polyhedral embedding of a net. *Vestnik Leningrad. Univ*, 15(7):66–80, 1960. In Russian.

5 Hooman Reisi Dehkordi, Fabrizio Frati, and Joachim Gudmundsson. Increasing-chord graphs on point sets. *J. Graph Algorithms Applications*, 19(2):761–778, 2015.

6 Erik D. Demaine and Joseph O'Rourke. *Geometric Folding Algorithms: Linkages, Origami, Polyhedra*. Cambridge University Press, 2007. `http://www.gfalop.org`.

7 Mohammad Ghomi. Affine unfoldings of convex polyhedra. *Geometry & Topology*, 18(5):3055–3090, 2014.

8 John M. Lee. *Riemannian Manifolds: An Introduction to Curvature*, volume 176. Springer Science & Business Media, 2006.

9 Anna Lubiw and Joseph O'Rourke. Angle-monotone paths in non-obtuse triangulations. In *Proc. 29th Canad. Conf. Comput. Geom.*, 2017. arXiv:1707.00219 [cs.CG]: `https://arxiv.org/abs/1707.00219`.

10 Joseph O'Rourke. Dürer's problem. In Marjorie Senechal, editor, *Shaping Space: Exploring Polyhedra in Nature, Art, and the Geometrical Imagination*, pages 77–86. Springer, 2013.

11 Joseph O'Rourke. Unfolding convex polyhedra via radially monotone cut trees. arXiv:1607.07421 [cs.CG]: `https://arxiv.org/abs/1607.07421`, 2016.

12 Joseph O'Rourke. Addendum to edge-unfolding nearly flat convex caps. arXiv:1709.02433 [cs.CG]: `http://arxiv.org/abs/1709.02433`, 2017.

13 Joseph O'Rourke. Edge-unfolding nearly flat convex caps. arXiv:1707.01006v2 [cs.CG]: `http://arxiv.org/abs/1707.01006`. Version 2, 2017.

14 Val Pincu. On the fewest nets problem for convex polyhedra. In *Proc. 19th Canad. Conf. Comput. Geom.*, pages 21–24, 2007.

A Crossing Lemma for Multigraphs

János Pach[1]
Ecole Polytechnique Fédérale de Lausanne and Rényi Institute, Hungarian Academy of Sciences
P.O.Box 127 Budapest, 1364, Hungary
pach@renyi.hu

Géza Tóth[2]
Rényi Institute, Hungarian Academy of Sciences
P.O.Box 127 Budapest, 1364, Hungary
geza@renyi.hu

Abstract

Let G be a drawing of a graph with n vertices and $e > 4n$ edges, in which no two adjacent edges cross and any pair of independent edges cross at most once. According to the celebrated Crossing Lemma of Ajtai, Chvátal, Newborn, Szemerédi and Leighton, the number of crossings in G is at least $c\frac{e^3}{n^2}$, for a suitable constant $c > 0$. In a seminal paper, Székely generalized this result to multigraphs, establishing the lower bound $c\frac{e^3}{mn^2}$, where m denotes the maximum multiplicity of an edge in G. We get rid of the dependence on m by showing that, as in the original Crossing Lemma, the number of crossings is at least $c'\frac{e^3}{n^2}$ for some $c' > 0$, provided that the "lens" enclosed by every pair of parallel edges in G contains at least one vertex. This settles a conjecture of Bekos, Kaufmann, and Raftopoulou.

2012 ACM Subject Classification Mathematics of computing → Graphs and surfaces

Keywords and phrases crossing number, Crossing Lemma, multigraph, separator theorem

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.65

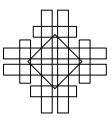
Related Version A related version of the paper is available at `https://arxiv.org/abs/1801.00721`.

Acknowledgements We are very grateful to Stefan Felsner, Michael Kaufmann, Vincenzo Roselli, Torsten Ueckerdt, and Pavel Valtr for their many valuable comments, suggestions, and for many interesting discussions during the Dagstuhl Seminar "Beyond-Planar Graphs: Algorithmics and Combinatorics", November 6-11, 2016, `http://www.dagstuhl.de/en/program/calendar/semhp/?semnr=16452`.

1 Introduction

A *drawing* of a graph G is a representation of G in the plane such that the vertices are represented by points, the edges are represented by simple continuous arcs connecting the corresponding pair of points without passing through any other point representing a vertex. In notation and terminology we do not make any distinction between a vertex (edge) and the point (resp., arc) representing it. Throughout this note we assume that any pair of edges intersect in finitely many points and no three edges pass through the same point. A common

[1] Supported by Swiss National Science Foundation Grants 200021-165977 and 200020-162884 and Schloss Dagstuhl – Leibniz Center for Informatics.
[2] Supported by National Research, Development and Innovation Office, NKFIH, K-111827 and Schloss Dagstuhl – Leibniz Center for Informatics.



interior point of two edges at which the first edge passes from one side of the second edge to the other, is called a *crossing*.

A very "successful concept for measuring non-planarity" of graphs is the *crossing number* of G [13], which is defined as the minimum number $\mathrm{cr}(G)$ of crossing points in any drawing of G in the plane. For many interesting variants of the crossing number, see [10], [8]. Computing $\mathrm{cr}(G)$ is an NP-hard problem [4], which is equivalent to the existential theory of reals [9].

The following statement, proved independently by Ajtai, Chvátal, Newborn, Szemerédi [1] and Leighton [6], gives a lower bound on the crossing number of a graph in terms of its number of vertices and number of edges.

▶ **Lemma** (Crossing Lemma, [1], [6]). *For any graph G with n vertices and $e > 4n$ edges, we have*

$$\mathrm{cr}(G) \geq \frac{1}{64}\frac{e^3}{n^2}.$$

Apart from the exact value of the constant, the order of magnitude of this bound cannot be improved. This lemma has many important applications, including simple proofs of the Szemerédi-Trotter theorem [14] on the maximum number of incidences between n points and n lines in the plane and of the best known upper bound on the number of halving lines induced by n points, due to Dey [3].

The same problem was also considered for *multigraphs* G, in which two vertices can be connected by several edges. As Székely [12] pointed out, if the *multiplicity* of an edge is at most m, that is, any pair of vertices of G is connected by at most m "parallel" edges, then the minimum number of crossings between the edges satisfies

$$\mathrm{cr}(G) \geq \frac{1}{64}\frac{e^3}{mn^2} \tag{1}$$

when $e \geq 4mn$. For $m = 1$, this gives the Crossing Lemma, but as m increases, the bound is getting weaker. It is not hard to see that this inequality is also tight up to a constant factor. Indeed, consider any (simple) graph with n vertices and roughly $e/m > 4n$ edges such that it can be drawn with at most $\frac{(e/m)^3}{n^2}$ crossings, and replace each edge by m parallel edges no pair of which share an interior point. The crossing number of the resulting multigraph cannot exceed $\frac{(e/m)^3}{n^2}m^2 = \frac{e^3}{mn^2}$.

It was suggested by Bekos, Kaufmann, and Raftopoulou [5] that the dependence on the multiplicity might be eliminated if we restrict our attention to a special class of drawings.

▶ **Definition 1.** A *drawing* of a multigraph G in the plane is called *branching*, or a *branching topological multigraph*, if the following conditions are satisfied.

- **(i)** If two edges are parallel (have the same endpoints), then there is at least one vertex in the interior and in the exterior of the simple closed curve formed by their union.
- **(ii)** If two edges share at least one endpoint, they cannot cross.
- **(iii)** If two edges do not share an endpoint, they can have at most one crossing.

Given a multigraph G, its *branching crossing number* is the smallest number $\mathrm{cr}_{\mathrm{br}}(G)$ of crossing points in any branching drawing of G. If G has no such drawing, set $\mathrm{cr}_{\mathrm{br}}(G) = \infty$.

According to this definition, $\mathrm{cr}_{\mathrm{br}}(G) \geq \mathrm{cr}(G)$ for every graph or multigraph G, and if G has no parallel edges, equality holds.

The main aim of this note is to settle the conjecture of Bekos, Kaufmann, and Raftopoulou.

▶ **Theorem 2.** *The branching crossing number of any multigraph G with n vertices and $e > 4n$ edges satisfies* $\mathrm{cr}_{\mathrm{br}}(G) \geq c\frac{e^3}{n^2}$*, for an absolute constant* $c > 10^{-7}$*.*

Unfortunately, the standard proofs of the Crossing Lemma by inductional or probabilistic arguments break down in this case, because the property that a drawing of G is branching is not hereditary: it can be destroyed by deleting vertices from G.

The bisection width of an *abstract* graph is usually defined as the minimum number of edges whose deletion separates the graph into two parts containing "roughly the same" number of vertices. In analogy to this, we introduce the following new parameter of *branching topological* multigraphs.

▶ **Definition 3.** The *branching bisection width* $\mathrm{b}_{\mathrm{br}}(G)$ of a *branching topological multigraph* G with n vertices is the minimum number of edges whose removal splits G into two *branching topological multigraphs*, G_1 and G_2, with no edge connecting them such that $|V(G_1)|, |V(G_2)| \geq n/5$.

A key element of the proof of Theorem 2 is the following statement establishing a relationship between the branching bisection width and the number of crossings of a branching topological multigraph.

▶ **Theorem 4.** *Let G be a branching topological multigraph with n vertices of degrees $d_1, d_2, \ldots, d_n$, and with $c(G)$ crossings. Then the branching bisection width of G satisfies*

$$\mathrm{b}_{\mathrm{br}}(G) \leq 22\sqrt{c(G) + \sum_{i=1}^{n} d_i^2 + n}.$$

By definition, the number of crossings $c(G)$ between the edges of G has to be at least as large as the branching crossing number of the abstract underlying multigraph of G.

To prove Theorem 2, we will use Theorem 4 recursively. Therefore, it is crucially important that in the definition of $\mathrm{b}_{\mathrm{br}}(G)$, both parts that G is cut into should be branching topological multigraphs themselves. If we are not careful, all vertices of $V(G)$ that lie in the interior (or in the exterior) of a closed curve formed by two parallel edges between $u, v \in G_1$, say, may end up in G_2. This would violate for G_1 condition (i) in the above definition of branching topological multigraphs. That is why the proof of Theorem 4 is far more delicate than the proof of the analogous statement for abstract graphs without multiple edges, obtained in [7].

For the proof of Theorem 2, we also need the following result.

▶ **Theorem 5.** *Let G be a branching topological multigraph with $n \geq 3$ vertices. Then the number of edges of G satisfies $e(G) \leq n(n-2)$, and this bound is tight.*

Our strategy for proving Theorem 2 is the following. Suppose, for a contradiction, that a multigraph G has a branching drawing in which the number of crossings is smaller than what is required by the theorem. According to Theorem 4, this implies that the branching bisection width of this drawing is small. Thus, we can cut the drawing into two smaller branching topological multigraphs, G_1 and G_2, by deleting relatively few edges. We repeat the same procedure for G_1 and G_2, and continue recursively until the size of every piece falls under a carefully chosen threshold. The total number of edges removed during this procedure is small, so that the small components altogether still contain a lot of edges. However, the number of edges in the small components is bounded from above by Theorem 5, which leads to the desired contradiction.

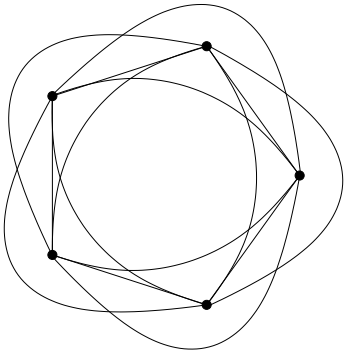


Figure 1 Theorem 5 is tight for every $n \geq 3$. Construction for $n = 5$.

Remarks.

1. Theorem 2 does not hold if we drop conditions (ii) and (iii) in the above definition, that is, if we allow two edges to cross more than once. To see this, suppose that n is a multiple of 3 and consider a tripartite topological multigraph G with $V(G) = V_1 \cup V_2 \cup V_3$, where all points of V_i belong to the line $x = i$ and we have $|V_i| = n/3$ for $i = 1, 2, 3$. Connect each point of V_1 to every point of V_3 by $n/3$ parallel edges: by one curve passing between any two (cyclically) consecutive vertices of V_2. We can draw these curves in such a way that any two edges cross at most twice, so that the number of edges is $e = e(G) = (n/3)^3$ and the total number of crossings is at most $2\binom{e}{2} < (n/3)^6$. On the other hand, the lower bound in Theorem 2 is $ce^3/n^2 > (c/3^9)n^7$, which is a contradiction if n is sufficiently large.
2. In the definition of *branching topological multigraphs*, for symmetry we assumed that the closed curve obtained by the concatenation of any pair of parallel edges in G has at least one vertex in its interior and at least one vertex in its exterior; see condition (i). It would have been sufficient to require that any such curve has at least one vertex in its *interior*, that is, any lens enclosed by two parallel edges contains a vertex. Indeed, by placing an isolated vertex v far away from the rest of the drawing, we can achieve that there is at least one vertex (namely, v) in the exterior of every lens, and apply Theorem 2 to the resulting graph with $n + 1$ vertices.
3. Throughout this paper, we assume for simplicity that a multigraph does not have *loops*, that is, there are no edges whose endpoints are the same. It is easy to see that Theorem 2, with a slightly worse constant c, also holds for topological multigraphs G having loops, provided that condition (ii) in the definition of branching topological multigraphs remains valid. In this case, one can argue that the total number of loops cannot exceed n. Subdividing every loop by an additional vertex, we get rid of all loops, and then we can apply Theorem 2 to the resulting multigraph of at most $2n$ vertices.

The rest of this note is organized as follows. In Section 2, we establish Theorem 5. In Section 3, we apply Theorems 4 and 5 to deduce Theorem 2. The proof of Theorem 4 is given in Section 4.

2 The number of edges in branching topological multigraphs and proof of Theorem 5

▶ **Lemma 6.** *Let G be a branching topological multigraph with $n \geq 3$ vertices and e edges, in which no two edges cross each other. Then $e \leq 3n - 6$.*

Proof. We can suppose without loss of generality that G is connected. Otherwise, we can achieve this by adding some edges of multiplicity 1, without violating conditions (i)-(iii) required for a drawing to be branching. We have a connected planar map with f faces, each of which is simply connected and has size at least 3. (The *size* of a face is the number of edges along its boundary, where an edge is counted twice if both of its sides belong to the face.) As in the case of simple graphs, we have that $2e$ is equal to the sum of the sizes of the faces, which is at least $3f$. Hence, by Euler's polyhedral formula,

$$2 = n - e + f \leq n - e + \frac{2}{3}e = n - \frac{1}{3}e,$$

and the result follows. ◀

▶ **Corollary 7.** *Let G be a branching topological multigraph with $n \geq 3$ vertices and e edges. Then for the number of crossings in G we have $c(G) \geq e - 3n + 6$.*

Proof. By our assumptions, each crossing belongs to precisely two edges. At each crossing, delete one of these two edges. The remaining topological graph G' has at least $e - c(G)$ edges. Since G' is a branching topological multigraph with no two crossing edges, we can apply Lemma 6 to obtain $e - c(G) \leq 3n - 6$. ◀

Proof of Theorem 5. Let G be a branching topological multigraph with n vertices. It is sufficient to show that for the degree of every vertex $v \in V(G)$ we have $d(v) \leq 2n - 4$. This implies that $e(G) \leq n(2n-4)/2 = n(n-2)$.

Let $v_1, v_2, \ldots, v_{n-1}$ denote the vertices of G different from v. Delete all edges of G that are not incident to v. No two remaining edges cross each other. If v is not adjacent to some $v_i \in V(G)$, then add a single edge vv_i without creating a crossing. The resulting topological multigraph, G', is also branching. Starting with any edge connecting v to v_1, list all edges incident to v in clockwise order, and for each edge write down its endpoint different from v. In this way, we obtain a sequence σ of length at least $d(v)$, consisting of the symbols $v_1, v_2, \ldots, v_{n-1}$, with possible repetition. Let σ' denote the sequence of length at least $d(v) + 1$ obtained from σ by adding an extra symbol v_1 at the end.

Property A: No two consecutive symbols of σ' are the same.

This is obvious for all but the last pair of symbols, otherwise the corresponding pair of edges of G' would form a simple closed Jordan curve with no vertex in its interior or in its exterior, contradicting the fact that G' is branching. The last two symbols of σ' cannot be the same either, because this would mean that σ starts and ends with v_1, and in the same way we arrive at a contradiction.

Property B: σ' does not contain a subsequence of the type $v_i \ldots v_j \ldots v_i \ldots v_j$ for $i \neq j$.

Indeed, otherwise the closed curve formed by the pair of edges connecting v to v_i would cross the closed curve formed by the pair of edges connecting v to v_j, contradicting the fact that G' is crossing-free.

A sequence with Properties A and B is called a *Davenport-Schinzel sequence of order 2*. It is known and easy to prove that any such sequence using $n-1$ distinct symbols has length at most $2n - 3$; see [11], page 6. Therefore, we have $d(v) + 1 \leq 2n - 3$, as required.

To see that the bound in Theorem 5 is tight, place a regular n-gon on the equator E (a great circle of a sphere), and connect any two consecutive vertices by a single circular arc

along E. Connect every pair of nonconsecutive vertices by two half-circles orthogonal to E: one in the Northern hemisphere and one in the Southern hemisphere. The total number of edges of the resulting drawing is $2\binom{n}{2} - n = n(n-2)$. See Fig. 1. ◀

3 Proof of Theorem 2 – using Theorems 4 and 5

Let G' be a branching topological multigraph of n' vertices and $e' > 4n'$ edges. If $e' \leq 10^8 n'$, then it follows from Corollary 7 that G' meets the requirements of Theorem 2.

To prove Theorem 2, suppose for contradiction that $e' > 10^8 n'$ and that the number of crossings in G' satisfies

$$c(G') < c(e')^3/(n')^2,$$

for a small constant $c > 0$ to be specified later.

Let d denote the average degree of the vertices of G', that is, $d = 2e'/n'$. For every vertex $v \in V(G)$ whose degree, $d(v)$, is larger than d, split v into several vertices of degree at most d, as follows. Let $vw_1, vw_2, \ldots, vw_{d(v)}$ be the edges incident to v, listed in clockwise order. Replace v by $\lceil d(v)/d \rceil$ new vertices, $v_1, v_2, \ldots, v_{\lceil d(v)/d \rceil}$, placed in clockwise order on a very small circle around v. By locally modifying the edges in a small neighborhood of v, connect w_j to v_i if and only if $d(i-1) < j \leq di$. Obviously, this can be done in such a way that we do not create any new crossing or two parallel edges that bound a region that contains no vertex. At the end of the procedure, we obtain a branching topological multigraph G with $e = e'$ edges, and $n < 2n'$ vertices, each of degree at most $d = 2e'/n' < 4e/n$.

Thus, for the number of crossings in G, we have

$$c(G) = c(G') < 4ce^3/n^2 \tag{2}$$

We break G into smaller components, according to the following procedure.

Decomposition Algorithm

Step 0. **Let** $G^0 = G, G_1^0 = G, M_0 = 1, m_0 = 1$.

Suppose that we have already executed Step i, and that the resulting branching topological graph, G^i, consists of M_i components, $G_1^i, G_2^i, \ldots, G_{M_i}^i$, each having at most $(4/5)^i n$ vertices. Assume without loss of generality that the first m_i components of G^i have at least $(4/5)^{i+1} n$ vertices and the remaining $M_i - m_i$ have fewer. Letting $n(G_j^i)$ denote the number of vertices of the component G_j^i, we have

$$(4/5)^{i+1} n(G) \leq n(G_j^i) \leq (4/5)^i n(G), \quad 1 \leq j \leq m_i. \tag{3}$$

Hence,

$$m_i \leq (5/4)^{i+1}. \tag{4}$$

Step $i+1$. **If**

$$(4/5)^i < \frac{1}{2} \cdot \frac{e}{n^2}, \tag{5}$$

then stop. (5) is called the *stopping rule*.

Else, for $j = 1, 2, \ldots, m_i$, delete $\mathrm{b}_{\mathrm{br}}(G_j^i)$ edges from G_j^i, as guaranteed by Theorem 4, such that G_j^i falls into two components, each of which is a branching topological graph with

at most $(4/5)n(G_j^i)$ vertices. Let G^{i+1} denote the resulting topological graph on the original set of n vertices. Clearly, each component of G^{i+1} has at most $(4/5)^{i+1}n$ vertices.

Suppose that the Decomposition Algorithm terminates in Step $k+1$. If $k>0$, then

$$(4/5)^k < \frac{1}{2}\cdot\frac{e}{n^2} \le (4/5)^{k-1}. \tag{6}$$

First, we give an upper bound on the total number of edges deleted from G. Using the fact that, for any nonnegative numbers $a_1, a_2, \ldots, a_m$,

$$\sum_{j=1}^{m}\sqrt{a_j} \le \sqrt{m\sum_{j=1}^{m}a_j}, \tag{7}$$

we obtain that, for any $0\le i<k$,

$$\sum_{j=1}^{m_i}\sqrt{c(G_j^i)} \le \sqrt{m_i\sum_{j=1}^{m_i}c(G_j^i)} \le \sqrt{(5/4)^{i+1}}\sqrt{c(G)} < \sqrt{(5/4)^{i+1}}\sqrt{4ce^3/n^2}.$$

Here, the last inequality follows from (2).

Denoting by $d(v,G_j^i)$ the degree of vertex v in G_j^i, in view of (7) and (4), we have

$$\begin{aligned}\sum_{j=1}^{m_i}\sqrt{\sum_{v\in V(G_j^i)}d^2(v,G_j^i)+n(G_j^i)} &\le \sqrt{m_i\left(\sum_{v\in V(G^i)}d^2(v,G^i)+n\right)}\\ &\le \sqrt{(5/4)^{i+1}}\sqrt{\max_{v\in V(G^i)}d(v,G^i)\cdot\sum_{v\in V(G^i)}d(v,G^i)+n}\\ &\le \sqrt{(5/4)^{i+1}}\sqrt{\frac{4e}{n}2e+n} < \sqrt{(5/4)^{i+1}}\frac{3e}{\sqrt{n}}.\end{aligned}$$

Thus, by Theorem 4, the total number of edges deleted during the decomposition procedure is

$$\begin{aligned}\sum_{i=0}^{k-1}\sum_{j=1}^{m_i}\mathrm{b_{br}}(G_j^i) &\le 22\sum_{i=0}^{k-1}\sum_{j=1}^{m_i}\sqrt{c(G_j^i)+\sum_{v\in V(G_j^i)}d^2(v,G_j^i)+n(G_j^i)}\\ &\le 22\sum_{i=0}^{k-1}\sum_{j=1}^{m_i}\sqrt{c(G_j^i)}+22\sum_{i=0}^{k-1}\sum_{j=1}^{m_i}\sqrt{\sum_{v\in V(G_j^i)}d^2(v,G_j^i)+n(G_j^i)}\\ &\le 22\left(\sum_{i=0}^{k-1}\sqrt{(5/4)^{i+1}}\right)\left(\sqrt{\frac{4ce^3}{n^2}}+\frac{3e}{\sqrt{n}}\right) < 350\frac{n}{\sqrt{e}}\left(\sqrt{\frac{4ce^3}{n^2}}+\frac{3e}{\sqrt{n}}\right)\\ &< 350(2\sqrt{ce}+3\sqrt{en}) < 350(2\sqrt{ce}+3\sqrt{e(2e/10^8)}) < \frac{e}{2},\end{aligned}$$

provided that $c\le 10^{-7}$. In the last line, we used our assumption that $e>10^8n'>(10^8/2)n$. The estimate for the term $\sum_{i=0}^{k-1}\sqrt{(5/4)^{i+1}}$ follows from (6).

So far we have proved that the number of edges of the graph G^k obtained in the final step of the Decomposition Algorithm satisfies

$$e(G^k) > \frac{e}{2}. \tag{8}$$

(Note that this inequality trivially holds if the algorithm terminates in the very first step, i.e., when $k = 0$.)

Next we give a lower bound on $e(G^k)$. The number of vertices of each connected component of G^k satisfies

$$n(G_j^k) \le (4/5)^k n < \frac{1}{2} \cdot \frac{e}{n^2} n = \frac{e}{2n}, \quad 1 \le j \le M_k.$$

By Theorem 5,

$$e(G_j^k) \le n^2(G_j^k) < n(G_j^k) \cdot \frac{e}{2n}.$$

Therefore, for the total number of edges of G^k we have

$$e(G^k) = \sum_{j=1}^{M_k} e(G_j^k) < \frac{e}{2n} \sum_{j=1}^{M_k} n(G_j^k) = \frac{e}{2},$$

contradicting (8). This completes the proof of Theorem 2. ◀

4 Branching bisection width vs. number of crossings – Proof of Theorem 4

Suppose that there is a weight function w on a set V. Then for any subset S of V, let $w(S)$ denote the total weight of the elements of S. We will apply the following separator theorem.

▶ **Lemma** (Separator Theorem, (Alon-Seymour-Thomas [2])). *Suppose that a graph G is drawn in the plane with no crossings. Let $V = \{v_1, \dots, v_n\}$ be the vertex set of G. Let w be a nonnegative weight function on V. Then there is a simple closed curve Φ with the following properties.*

- **(i)** *Φ meets G only in vertices.*
- **(ii)** $|\Phi \cap V| \le 3\sqrt{n}$
- **(iii)** *Φ divides the plane into two regions, D_1 and D_2, let $V_i = D_i \cap V$. Then for $i = 1, 2$,*

$$w(V_i) + \frac{1}{2} w(\Phi \cap V) \le \frac{2}{3} w(V).$$

Consider a branching drawing of G with exactly $c(G) = \mathrm{cr}_{\mathrm{br}}(G)$ crossings. Let V_0 be the set of *isolated* vertices of G, and let $v_1, v_2, \dots, v_m$ be the other vertices of G with degrees $d_1, d_2, \dots, d_m$, respectively. Introduce a new vertex at each crossing. Denote the set of these vertices by V_X.

For $i = 1, 2 \dots, m$, replace vertex v_i by a set V_i of vertices forming a very small $d_i \times d_i$ piece of a square grid, in which each vertex is connected to its horizontal and vertical neighbors. Let each edge incident to v_i be hooked up to distinct vertices along one side of the boundary of V_i without creating any crossing. These d_i vertices will be called the *special boundary vertices* of V_i.

Note that we modified the drawing of the edges only in small neighborhoods of the grids V_i, that is, in nonoverlapping small neighborhoods of the vertices of G, far from any crossing.

Thus, we obtain a (simple) topological graph H, of $|V_X| + \sum_{i=0}^{m} |V_i| \le c(G) + \sum_{i=1}^{m} d_i^2 + n$ vertices and with no crossing; see Fig. 2. For every $1 \le i \le m$, assign weight $1/d_i$ to each special boundary vertex of V_i. Assign weight 1 to every vertex of V_0 and weight 0 to all other vertices of H. Then $w(V_i) = 1$ for every $1 \le i \le n$ and $w(v) = 1$ for every $v \in V_0$. Consequently, $w(V(H)) = n$.

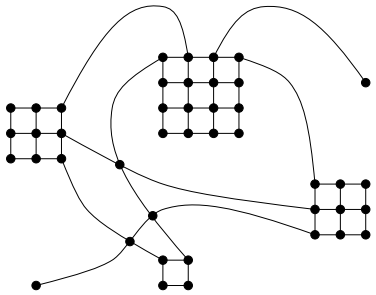


Figure 2 Topological graph H.

Apply the Separator Theorem to H. Let Φ denote the closed curve satisfying the conditions of the theorem. Let $A(\Phi)$ and $B(\Phi)$ denote the region *interior* and the *exterior* of Φ, respectively. For $1 \leq i \leq m$, let $A_i = V_i \cap A(\Phi)$, $B_i = V_i \cap B(\Phi)$, $C_i = V_i \cap \Phi$. Finally, let $C_X = V_X \cap \Phi$.

▶ **Definition 8.** For any $1 \leq i \leq m$, we say that
- V_i is of *type* A if $w(A_i) \geq \frac{5}{6}$,
- V_i is of *type* B if $w(B_i) \geq \frac{5}{6}$,
- V_i is of *type* C, otherwise.

For every $v \in V_0$,
- v is of *type* A if $v \in A(\Phi)$,
- v is of *type* B if $v \in B(\Phi)$,
- v is of *type* C, if $v \in \Phi$.

Define a partition $V(G) = V_A \cup V_B$ of the vertex set of G, as follows. For any $1 \leq i \leq m$, let $v_i \in V_A$ (resp. $v_i \in V_B$) if V_i is of type A (resp. type B). Similarly, for every $v \in V_0$, let $v \in V_A$ (resp. $v \in V_B$) if v is of type A (resp. type B). The remaining vertices will be assigned either to V_A or to V_B so as to minimize $\big||V_A| - |V_B|\big|$.

▶ **Claim 9.** $\frac{n}{5} \leq |V_A|, |V_B| \leq \frac{4n}{5}$

Proof. To prove the claim, define another partition $V(H) = \overline{A} \cup \overline{B} \cup \overline{C}$ such that $\overline{A} \cap V_i = A \cap V_i$ and $\overline{B} \cap V_i = B \cap V_i$ for V_0 and for every V_i of type C. If V_i is of type A (resp. type B), then let $V_i = \overline{A}_i \subset \overline{A}$ (resp. $V_i = \overline{B}_i \subset \overline{B}$), finally, let $\overline{C} = V(H) - \overline{A} - \overline{B}$.

For any V_i of type A, we have $w(\overline{A}_i) - w(A_i) \leq \frac{w(A_i)}{5}$. Similarly, for any V_i of type B, we have $w(\overline{B}_i) - w(B_i) \leq \frac{w(B_i)}{5}$. Therefore,

$$|w(\overline{A}) - w(A)| \leq \frac{1}{5} \cdot \max\{w(A), w(B)\} \leq \frac{2n}{15}.$$

Hence, $\frac{n}{5} \leq w(\overline{A}) \leq \frac{4n}{5}$ and, analogously, $\frac{n}{5} \leq w(\overline{B}) \leq \frac{4n}{5}$. In particular, $|w(\overline{A}) - w(\overline{B})| \leq \frac{3n}{5}$. Using the minimality of $\big||V_A| - |V_B|\big|$, we obtain that $\big||V_A| - |V_B|\big| \leq \frac{3n}{5}$, which implies Claim 9. ◀

▶ **Claim 10.** *For any* $1 \leq i \leq n$,
- **(i)** *if* V_i *is of type* A *(resp. of type* B*), then* $|C_i| \geq w(B_i)d_i$ *(resp.* $|C_i| \geq w(A_i)d_i$*);*
- **(ii)** *if* V_i *is of type* C*, then* $|C_i| \geq \frac{d_i}{6}$.

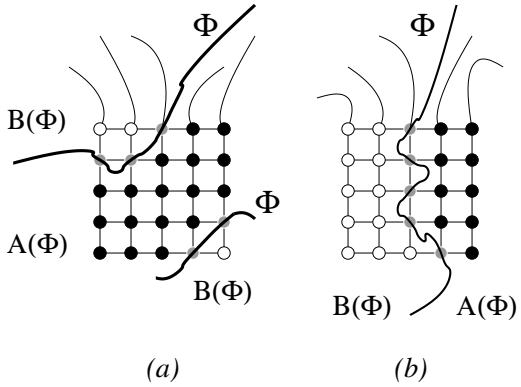


Figure 3 Parts (a) and (b) show a grid of type A and C, respectively.

Proof. In V_i, every connected component belonging to A_i is separated from every connected component belonging to B_i by vertices in C_i. There are $w(A_i)d_i$ (resp. $w(B_i)d_i$) special boundary vertices in V_i, which belong to A_i (resp. B_i). It can be shown by an easy case analysis that the number of separating points $|C_i| \geq \min\{w(A_i), w(B_i)\}d_i$, and Claim 10 follows; see Fig. 3. ◀

▶ **Claim 11.** *Let $V = V(G)$. There is a closed curve Ψ, not passing through any vertex of H, whose interior and exterior are denoted by $A(\Psi)$ and $B(\Psi)$, resp., such that*

(i) $V \cap A(\Psi) = V_A$,

(ii) $V \cap B(\Psi) = V_B$,

(iii) *the total number of edges of G intersected by Ψ is at most*

$$18\sqrt{c(G) + \sum_{i=1}^{n} d_i^2 + n}.$$

Proof. For any $1 \leq i \leq m$, we say that

- V_i is of *type 1* if $|C_i| \geq d_i/6$,
- V_i is of *type 2* if $|C_i| < d_i/6$.

For every $v \in V_0$,

- v is of *type 1* if $v \in \Phi$,
- v is of *type 2* if $v \in A(\Phi) \cup B(\Phi)$.

It follows from Claim 10 that if a set V_i or an isolated vertex $v \in V_0$ is of type C, then it is also of type 1.

Next, we modify the curve Φ in small neighborhoods of the grids V_i and of the isolated vertices $v \in V_0$ to make sure that the resulting curve Ψ satisfies the conditions in the claim.

Assume for simplicity that $v_i \in V_A$; the case $v_i \in V_B$ can be treated analogously. If v_i is a vertex of degree at most 1 and Φ passes through v_i, slightly perturb Φ in a small neighborhood of v_i (or slightly shift v_i) so that after this change v_i lies in the interior of Φ. Suppose next that the degree of v_i is at least 2. Let S_i and $S_i' \subset S_i$ be two closed squares containing V_i in their interiors, and assume that S_i (and, hence, S_i') is only slightly larger than the convex hull of the vertices of V_i. We distinguish two cases.

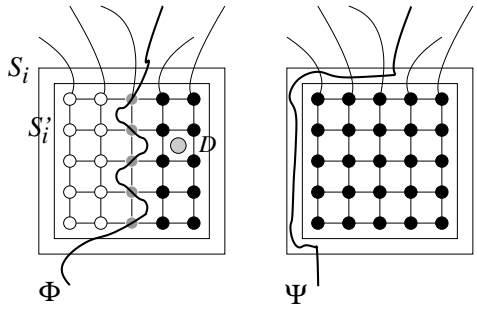


Figure 4 Claim 11, Case 1.

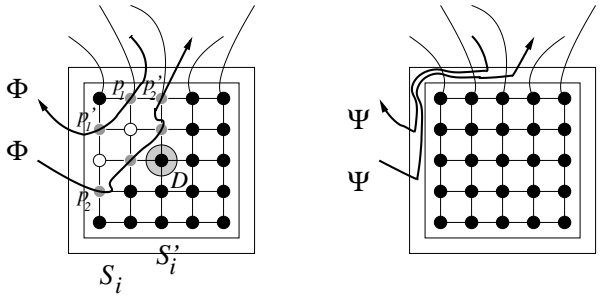


Figure 5 Claim 11, Case 2.

Case 1. *V_i is of type 1.* Let D be a small disk in S_i' that belongs to the interior of Φ and let p be its center. Let $\tau : S_i \to S_i$ be a homeomorphism of S_i to itself which keeps the boundary of S_i fixed and let $\tau(D) = S_i'$. Observe that every piece of Φ within the convex hull of the vertices of V_i is mapped into an arc in the very narrow ring $S_i \setminus S_i'$. In particular, if we keep the vertices and the edges of the grid $H[V_i]$ (as well as all other parts of the drawing) fixed, after this local modification Φ will avoid all vertices of V_i and it may intersect only those (at most d_i) edges incident to V_i which correspond to original edges of G and end at some special boundary vertex of V_i. Moreover, after this modification, every vertex of V_i will lie in $A(\Phi)$, in the *interior* of Φ.

Case 2. *V_i is of type 2.* In this case, by Claim 10, V_i is of type A.

Orient Φ arbitrarily. Let $(p_1, p_1'), (p_2, p_2'), \ldots$ denote the point pairs at which Φ enters and leaves the convex hull of V_i, so that the arc between $p_j p_j'$ lies inside the convex hull of V_i, for every j. Note that both p_j and p_j' are vertices of V_i. In view of the fact that $|C_i| \leq d_i/6$, we know that the (graph) distance between p_j and p_j' (in $H[V_i]$) is at most $d_i/6$. More precisely, for every j, the points p_j and p_j' divide the boundary of the convex hull of V_i into two arcs. We call the shorter of these arcs the *boundary interval defined by p_j and p_j'*, and denote it by $[p_j, p_j']$. By assumption, the *length* of $[p_j, p_j']$. the number of edges of $H[V_i]$ comprising $[p_j, p_j']$, is at most $d_i/6$.

It is not hard to see that the curve Φ cannot came close to the center p of V_i and that p belongs to the interior of Φ. Let D be a small disk centered at p. Then D also belongs to the interior of Φ. Let $\tau : S_i \to S_i$ be a homeomorphism of S_i to itself such that (i) τ keeps the boundary of S_i fixed, (ii) $\tau(D) = S_i'$, (iii) $\tau(p) = p$, and (iv) for any $q \in S_i$, that points p, q, and $\tau(q)$ are collinear. Observe that every piece (p_j, p_j'), of Φ within the convex hull of the vertices of V_i is mapped into an arc in the very narrow ring $S_i \setminus S_i'$, along the

corresponding boundary interval, $[p_j, p'_j]$, defined by p_j and p'_j. In particular, if we keep the vertices and edges of the grid $H[V_i]$ (as well as all other parts of the drawing) fixed, after this local modification Φ will avoid all vertices of V_i and it may intersect only those (at most $d_i/6$) edges incident to V_i which correspond to original edges of G and end at some special boundary vertex of V_i in a boundary interval. Moreover, now every vertex of V_i will lie *inside* Φ.

Repeat the above local modification for each V_i and for each $v \in V_0$. The resulting curve, Ψ, satisfies conditions (i) and (ii). It remains to show that it also satisfies (iii).

To see this, denote by E_X the set of all edges of H adjacent to at least one element of C_X. For any $1 \leq i \leq m$, define $E_i \subset E(H)$ as follows. If V_i is of type 1, then let all edges of H leaving V_i belong to E_i. If V_i is of type 2, then by Claim 10, it can be of type A or B, but not C. Let E_i consist of all edges leaving V_i and crossed by Ψ.

For any $1 \leq i \leq m$, let E'_i denote the set of edges of G corresponding to the elements of E_i $(0 \leq i \leq m)$ and let E'_X denote the set of edges corresponding to the elements of E_X.

Clearly, we have $|E'_i| \leq |E_i|$, because distinct edges of G give rise to distinct edges of H. Since V_A and V_B are on different sides of Ψ, it crosses all edges between V_A and V_B.

Obviously, $|E'_X| \leq |E_X| \leq 4|C_X|$. By Claim 10, if V_i is of type 1, then $|E'_i| \leq |E_i| = d_i \leq 6|C_i|$. If V_i is of type 2, then $|E'_i| \leq |E_i| = d_i \leq |C_i|$. Therefore,

$$|E(V_A, V_B)| \leq |\cup_{i=0}^{n} E'_i| \leq \sum_{i=0}^{n} |E_i| \leq 6|C| \leq 18\sqrt{c(G) + \sum_{i=1}^{n} d_i^2 + n}.$$

This finishes the proof of Claim 11. ◀

Now we are in a position to complete the proof of Theorem 4. Remove those edges of G that are cut by Ψ. Let G_A (resp. G_B) be the subgraph of the resulting graph G', induced by V_A (resp. V_B), with the inherited drawing. Suppose that, e.g., G_B is not a branching topological graph. Then it has an *empty lens*, that is, a region bounded by two parallel edges that does not contain any vertex of V_B. There are two types of empty lenses: bounded and unbounded. We show that there are at most $\sqrt{c(G)}$ bounded empty lenses, and at most $\sqrt{c(G)}$ unbounded empty lenses in G_B.

Suppose that e and e' are two parallel edges between v and v' which enclose a bounded empty lens L. Then v and v' are in the exterior of Ψ, and Ψ does not cross the edges e and e'. As G was a branching topological multigraph, both L and its complement contain at least one vertex of G in their interiors. Since L is empty in G_B, it follows that all vertices of G inside L must belong to V_A, and, hence, must lie in the interior of Ψ. Thus, Ψ must lie entirely inside the lens L.

Suppose now that f and f' are two other parallel edges between two vertices u and u', and they determine another bounded empty lens M. Arguing as above, we obtain that Ψ must also lie entirely inside M. Then v and v' are outside of M, and u and u' are outside of L. Therefore, these four edges determine four crossings. Any such crossing can belong to only one pair of bounded empty lenses $\{L, M\}$, we conclude that for the number of bounded empty lenses k in G_B we have $4\binom{f}{2} \leq c(G)$, therefore, $k \leq \sqrt{c(G)}$. Analogously, there are at most $\sqrt{c(G)}$ unbounded empty lenses in G_B.

We can argue in exactly the same way for G_A. Thus, altogether there are at most $4\sqrt{c(G)}$ empty lenses in G_A and G_B. If we delete a boundary edge of each of them, then no empty lens is left.

Thus, by deleting the edges of G crossed by Ψ and then one boundary edge of each empty lens, we obtain a decomposition of G into two branching topological multigraphs, and the number of deleted edges is at most

$$18\sqrt{c(G)+\sum_{i=1}^{n} d_i^2+n}+4\sqrt{c(G)} \leq 22\sqrt{c(G)+\sum_{i=1}^{n} d_i^2+n}.$$

This concludes the proof of Theorem 4. ◀

References

1. Miklós Ajtai, Vašek Chvátal, Monroe M Newborn, and Endre Szemerédi. Crossing-free subgraphs. *North-Holland Mathematics Studies*, 60(C):9–12, 1982.
2. Noga Alon, Paul Seymour, and Robin Thomas. Planar separators. *SIAM Journal on Discrete Mathematics*, 7(2):184–193, 1994.
3. Tamal K Dey. Improved bounds for planar k-sets and related problems. *Discrete & Computational Geometry*, 19(3):373–382, 1998.
4. Michael R Garey and David S Johnson. Crossing number is np-complete. *SIAM Journal on Algebraic Discrete Methods*, 4(3):312–316, 1983.
5. M. Kaufmann. Personal communication. Beyond-Planar Graphs: Algorithmics and Combinatorics, Schloss Dagstuhl, Germany, November 6–11, 2016.
6. Frank Thomson Leighton. *Complexity issues in VLSI: optimal layouts for the shuffle-exchange graph and other networks*. MIT press, Cambridge, 1983.
7. J Pach, F Shahrokhi, and M Szegedy. Applications of the crossing number. *Algorithmica*, 16(1):111–117, 1996.
8. János Pach and Géza Tóth. Thirteen problems on crossing numbers. *Geombinatorics*, 9:199–207, 2000.
9. Marcus Schaefer. Complexity of some geometric and topological problems. In *International Symposium on Graph Drawing, Lecture Notes in Computer Science*, volume 5849, pages 334–344. Springer, 2010.
10. Marcus Schaefer. The graph crossing number and its variants: A survey. *The electronic journal of combinatorics*, 1000:DS21, 2013.
11. Micha Sharir and Pankaj K Agarwal. *Davenport-Schinzel sequences and their geometric applications*. Cambridge University Press, 1995.
12. László A Székely. Crossing numbers and hard erdős problems in discrete geometry. *Combinatorics, Probability and Computing*, 6(3):353–358, 1997.
13. László A Székely. A successful concept for measuring non-planarity of graphs: the crossing number. *Discrete Mathematics*, 276(1-3):331–352, 2004.
14. Endre Szemerédi and William T. Trotter. Extremal problems in discrete geometry. *Combinatorica*, 3(3-4):381–392, 1983.

Near-Optimal Coresets of Kernel Density Estimates

Jeff M. Phillips[1]
School of Computing, University of Utah
Salt Lake City, USA
jeffp@cs.utah.edu

Wai Ming Tai
School of Computing, University of Utah
Salt Lake City, USA
wmtai@cs.utah.edu

Abstract

We construct near-optimal coresets for kernel density estimate for points in $\mathbb{R}^d$ when the kernel is positive definite. Specifically we show a polynomial time construction for a coreset of size $O(\sqrt{d\log(1/\varepsilon)}/\varepsilon)$, and we show a near-matching lower bound of size $\Omega(\sqrt{d}/\varepsilon)$. The upper bound is a polynomial in $1/\varepsilon$ improvement when $d \in [3, 1/\varepsilon^2)$ (for all kernels except the Gaussian kernel which had a previous upper bound of $O((1/\varepsilon)\log^d(1/\varepsilon))$) and the lower bound is the first known lower bound to depend on d for this problem. Moreover, the upper bound restriction that the kernel is positive definite is significant in that it applies to a wide-variety of kernels, specifically those most important for machine learning. This includes kernels for information distances and the sinc kernel which can be negative.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Coresets, Kernel Density Estimate, Discrepancy

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.66

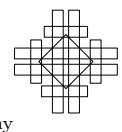
1 Introduction

Kernel density estimates are pervasive objects in data analysis. They are the classic way to estimate a continuous distribution from a finite sample of points [28, 27]. With some negative weights, they are the prediction function in kernel SVM classifiers [25]. They are the core of many robust topological reconstruction approaches [22, 12, 6]. And they arise in many other applications including mode estimation [1], outlier detection [26], regression [11], and clustering [23].

Generically, consider a dataset $P \subset \mathbb{R}^d$ of size n, and a kernel $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, for instance the Gaussian kernel $K(x,p) = \exp(-\alpha^2\|x-p\|^2)$ with $1/\alpha$ as a bandwidth parameter. Then a kernel density estimate is defined at any point $x \in \mathbb{R}^d$ as $\text{KDE}_P(x) = \frac{1}{n}\sum_{p\in P} K(x,p)$.

Given that it takes $O(n)$ time to evaluate KDE_P, and that data sets are growing to massive sizes, in order to continue to use these powerful modeling objects, a common approach is to replace P with a much smaller data sets Q so that KDE_Q approximates KDE_P. While statisticians have classically studied various sorts of average deviations (L_2 [28, 27] or L_1 error [9]), for most modern data modeling purposes, a worst-case L_∞ is more relevant (e.g., for preserving classification margins [25], density estimates [30], topology [22], and hypothesis

[1] Thanks to supported by NSF CCF-1350888, IIS-1251019, ACI-1443046, CNS-1514520, and CNS-1564287



testing on distributions [13]). Specifically this error guarantee preserves

$$\| \operatorname{KDE}_P - \operatorname{KDE}_Q \|_\infty = \max_{x \in \mathbb{R}^d} | \operatorname{KDE}_P(x) - \operatorname{KDE}_Q(x)| \leq \varepsilon.$$

We call such a set Q an *ε-KDE coreset* of P.

Traditionally the approximate set Q has been considered to be constructed as a random sample of P [28, 27, 16], sometimes known as a Nyström approximation [10]. However, in the last decade, a slew of data-aware approaches have been developed that can obtain a set Q with the same L_∞ error guarantee, but with considerably smaller size.

To describe either the random sample results or the data-aware approaches, we first need to be more specific about the properties of the kernel functions. We start with *positive definite* kernels, the central class required for most machine learning approaches to work [15].

Table 1 Example Positive Definite Kernels

	$K(x,p) =$	domain
Gaussian	$\exp(-\alpha^2\|x-p\|^2)$	$\mathbb{R}^d$
Laplacian	$\exp(-\alpha\|x-p\|)$	$\mathbb{R}^d$
Exponential	$\exp(-\alpha(1-\langle x,p\rangle))$	$\mathbb{S}^d$
JS	$\exp(-\alpha(H(\frac{x+p}{2}) - \frac{H(x)+H(p)}{2}))$	Δ^d
Helinger	$\exp(-\alpha\sum_{i=1}^d(\sqrt{x_i}-\sqrt{p_i})^2)$	Δ^d
Sinc	$\frac{\sin(\alpha\|x-p\|)}{\alpha\|x-p\|}$	$\mathbb{R}^{d\leq 3}$

Postive definite kernels. Consider a kernel $K : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$ defined over some domain $\mathcal{D}$ (often $\mathbb{R}^d$). It is called a *positive definite kernel* if, for any m points $x_1, x_2, \ldots, x_m \in \mathcal{D}$, they are used to define an $m \times m$ Gram matrix G so each i, j entry is defined $G_{i,j} = K(x_i, x_j)$, and the matrix G is positive definite. Recall, a matrix G is positive definite if any vector $z \in \mathbb{R}^n$ that is not all zeros satisfies $z^T G z > 0$. Moreover, a positive definite matrix G can always be decomposed as a product $H^T H$ with real-valued matrix H.

There are many positive definite kernels, and we will next highlight a few. We normalize all kernels so $K(x, x) = 1$ for all $x \in \mathcal{D}$ and therefore $|K(x, y)| \leq 1$ for all $x, y \in \mathcal{D}$. We will use $\alpha > 0$ as a parameter, where $1/\alpha$ represents the bandwidth, or smoothness of the kernel. For $\mathcal{D} = \mathbb{R}^d$ the most common positive definite kernels [29] are the Gaussian (described earlier) and the Laplacian, defined $\exp(-\alpha\|x - y\|)$ for $x, y \in \mathbb{R}^d$. Another common domain is $\Delta^d = \{x \in \mathbb{R}^{d+1} \mid \sum_{i=1}^d x_i = 1,\ x_i \geq 0\}$, for instance in representing discrete distributions such as normalized counts of words in a text corpus or fractions of tweets per geographic region. Common positive definite kernels for $x, y \in \Delta^d$ include the Hellinger kernel $\exp(-\alpha\sum_{i=1}^d(\sqrt{x_i} - \sqrt{y_i})^2)$ and the Jensen-Shannon (JS) divergence kernel $\exp(-\alpha(H(\frac{x+y}{2}) - \frac{H(x)+H(y)}{2}))$, where $H(x) = \sum_{i=1}^d -x_i \log x_i$ is entropy [14]. In other settings it is more common to normalize data points x to lie on a sphere $\mathbb{S}^d = \{x \in \mathbb{R}^{d+1} \mid \|x\| = 1\}$. Then with $x, y \in \mathbb{S}^d$, the exponential kernel $\exp(-\alpha(1 - \langle x, y\rangle))$ is positive definite [15]. Perhaps surprisingly, positive definite kernels do not need to satisfy $K(x, y) \geq 0$. For $x, y \in \mathbb{R}^d$, the sinc kernel is defined as $\frac{\sin(\alpha\|x-y\|)}{\alpha\|x-y\|}$ and is positive definite for $d = \{1, 2, 3\}$ [24].

Other classes of kernels. There are other ways to characterize kernels, which provide sufficient conditions for various other coreset bounds. For clarity, we describe these for kernels with a domain of $\mathbb{R}^d$, but they can apply more generally.

We say a kernel K is *C_K-Lipschitz* if, for any $x, y, z \in \mathbb{R}^d$, $|K(x, z) - K(y, z)| \leq C_k \cdot \|x - y\|$. This ensures that the kernels do not fluctuate too widely, a necessity for robustness, but also prohibits "binary" kernels; for instance the *ball kernel* is defined $K(x, y) = \{1$ if $\|x - y\| \leq r$; and 0 otherwise$\}$. Such binary kernels are basically range counting queries (for instance

the ball kernel corresponds with a range defined by a ball), and as we will see, this distinction allows the bounds for ε-KDE coresets to surpass lower bounds for coresets for range counting queries. Aside from the ball kernel, all kernels we discuss in this paper will be C_K-Lipshcitz.

Another way to characterize a kernel is with their shape. We can measure this by considering binary ranges defined by super-level sets of kernels. For instance, given a fixed K and $x \in \mathbb{R}^d$, and a threshold $\tau \in \mathbb{R}$, the *super-level set* is $\{p \in \mathbb{R}^d \mid K(x,p) \geq \tau\}$. For a fixed K, the family of such sets over all choices of x and τ describes a range space with ground set $\mathbb{R}^d$. For many kernels the VC-dimension of this range space is bounded; in particular, for common kernels, this range is equivalent to those defined by balls in $\mathbb{R}^d$. Notably, the sinc kernel, which is positive-definite for $\mathbb{R}^d$ with $d \leq 3$ does not correspond to a range space with bounded VC-dimension.

Finally, we mention that kernels being *characteristic* [29] is an important property for many bounds that rely on mappings from data points $x \in \mathbb{R}^d$ to a function space $K(x,\cdot) = \phi_K(x)$. A characteristic kernel requires that the kernel K is positive definite, and the mapping $\phi_K(x)$ is isomorphic and hence its induced distance $D_K(p,x) = \sqrt{\|\phi_K(x)\|_K^2 + \|\phi_K(p)\|_K^2 - 2\langle \phi_K(p), \phi_K(x)\rangle_K}$ is a metric.

Discrepancy-based approaches. Our approach for creating an ε-KDE coreset will follow a technique for creating range counting coresets [8, 19, 5]. It focuses on assigning a coloring $\chi : P \to \{-1, +1\}$ to P. Then retains either all $P_+ = \{p \in P \mid \chi(p) = +1\}$ or the remainder P_-, and recursively applies this halving until a small enough coreset Q has been retained.

Classically, when the goal is to compute a range counting coreset for a range space $(P, \mathcal{R})$, then the specific goal of the coloring is to minimize discrepancy

$$\mathsf{disc}_R(P,\chi) = \left| \sum_{p \in P \cap R} \chi(p) \right|$$

over all choices of ranges $R \in \mathcal{R}$. In the KDE-setting we consider a *kernel range space* $(P, \mathcal{K})$ where $\mathcal{K} = \{K(x,\cdot) \mid x \in \mathcal{D}\}$ defined by kernel $K : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$ and a fixed domain $\mathcal{D}$ which is typically assumed, and usually $\mathcal{D} = \mathbb{R}^d$. We instead want to minimize the kernel discrepancy

$$\mathsf{disc}(P,\chi,x) = \left| \sum_{p \in P} \chi(p) K(x,p) \right|.$$

Now in contrast to the case with the binary range space $(P, \mathcal{R})$, *each* point $p \in P$ is *partially* inside the "range" where the amount inside is controlled by the kernel K. Understanding the quantity

$$\mathsf{disc}(n, \mathcal{K}) = \max_{P : |P| = n} \min_{\chi} \max_{x \in \mathcal{D}} \mathsf{disc}(P, \chi, x)$$

is key. If for a particular $\mathcal{K}$ we have $\mathsf{disc}(n,\mathcal{K}) = n^\tau$ or $\mathsf{disc}(n,\mathcal{K}) = \log^\eta n$, then applying the recursive halving algorithm obtains an ε-KDE coreset of size $O(1/\varepsilon^{1/(1-\tau)})$ and $O((1/\varepsilon)\log^\eta(1/\varepsilon))$, respectively [20].

Previous results. We now highlight some previous size bounds for ε-KDE coresets as shown in Table 2; see also Phillips and Tai [21] for a more thorough review of the historical progress on this problem. For simplicity, we report the size when only a constant probability of failure bound is required for randomized algorithms.

For kernels with bounded VC-dimension of their super-level sets, $O(d/\varepsilon^2)$ random samples are sufficient to create and ε-KDE coreset [16]. For characteristic kernels [13] a bound of $O(1/\varepsilon^4)$, independent of dimension is possible.

Table 2 Asymptotic ε-KDE coreset sizes in terms of ε and d. SISS = Shift- and rotation invariant, and somewhere-steep (see Section 3).

Paper	Coreset Size	Restrictions and Notes
Joshi *et al.* [16]	d/ε^2	bounded VC; random sample
Gretton *et al.* [13]	$1/\varepsilon^4$	characteristic kernels; random sample
Bach *et al.* [2, 21]	$1/\varepsilon^2$	characteristic kernels; greedy
Phillips [20]	$(1/\varepsilon)^{\frac{2d}{d+2}} \log^{\frac{d}{d+2}}(1/\varepsilon)$	Lipschitz, d is constant; discrepancy-based
Phillips [20]	$\Theta(1/\varepsilon)$	$d = 1$
Phillips and Tai [21]	$(1/\varepsilon)\log^d(1/\varepsilon)$	Gaussian, d is constant; discrepancy-based
Phillips and Tai [21]	$\Omega(1/\varepsilon^2)$	SISS (e.g., Gaussian); $d = \Omega(1/\varepsilon^2)$

Then Bach *et al.* [2] showed that a greedy approach in the function space (based on the Frank-Wolfe technique) can obtain a bound of $O(1/\varepsilon^2)$, independent of dimension. This created a non-uniformly weighted coreset, and Phillips and Tai [21] provided a construction and analysis with uniform weighting.

Joshi *et al.* [16] first showed that sub-$1/\varepsilon^2$ is possible using a discrepancy-based approach. These bounds were superseded by Phillips [20] that presented a size bound of $O(((1/\varepsilon^2)\log(1/\varepsilon))^{d/(d+2)})$ for constant d, or $O(1/\varepsilon)$ for $d = 1$. For $d = 2$, this bound is near-linear in $1/\varepsilon$, specifically $O((1/\varepsilon)\sqrt{\log(1/\varepsilon)})$ and matches our bound. He also showed a lower bound of $\Omega(1/\varepsilon)$. Notably, this upper bound for constant d is strictly smaller than those of the binary range space for balls, based on VC-dimension.

Then recently, Phillips and Tai [21] greatly improved the upper bound to $O((1/\varepsilon)\log^d(1/\varepsilon))$ also based on discrepancy-based approaches, but this only applies to the Gaussian kernel. It is based on a special decomposable structure of the Gaussian, and does not seem to generalize. They also provided a lower bound of $\Omega(1/\varepsilon^2)$, but this construction required $d = \Omega(1/\varepsilon^2)$ dimensions, so does not imply much in the low-dimensional setting.

1.1 Our Results

We first show a new upper bound on the size of an ε-KDE coreset of $O((1/\varepsilon)\sqrt{d\log(1/\varepsilon)})$ in Section 2. The main restriction on the kernel K is that it is positive definite, a weaker bound than the similar characteristic assumption. There are also fairly benign restrictions that K is Lipschitz and is value greater than $1/|P|$ (or $\geq \varepsilon^2$) for a bounded region due to the specifics of some geometric preprocessing. Noteably, this upper bound applies to a very wide range of kernels including the sinc kernel, whose super-level sets do not have bounded VC-dimension and is not characteristic, so no non-trivial ε-KDE coreset bound was previously known. Moreover, unlike previous discrepancy-based approaches, we do not need to assume the dimension d is constant.

We then show a nearly-matching lower bound on the size of an ε-KDE coreset of $\Omega(\sqrt{d}/\varepsilon)$, in Section 3. This construction requires a standard restriction that it is shift- and rotation-invariant, and a benign one that it is somewhere-steep (see Section 3), satisfied by all common kernels. This closes the problem for many kernels (e.g., Gaussians, Laplace), except for a $\sqrt{\log(1/\varepsilon)}$ factor when $1 < d < 1/\varepsilon^2$. The gap filled by the new bounds are shown in Table 3.

Table 3 Size bounds for ε-KDE coresets for Gaussian and Laplace kernels; also holds under more general assumption, see text. ($\star$) For $d = 2$, [20] matches upper bound.

d	Upper	Lower	
1	$1/\varepsilon$	$1/\varepsilon$	[20]
$[2, \frac{1}{\varepsilon^2})$	$\sqrt{d}/\varepsilon \cdot \sqrt{\log\frac{1}{\varepsilon}}$	$\sqrt{d}/\varepsilon$	**new***
$\geq \frac{1}{\varepsilon^2}$	$1/\varepsilon^2$	$1/\varepsilon^2$	[2, 21]

Our approach and context. As mentioned above, bounding the size ε-KDE coresets can be reduced to bounding kernel discrepancy. The range space discrepancy problem, for a range space $(P, \mathcal{R})$, has been widely studied in multiple areas [17, 7]. For instance, Tusnady's problem restricts $\mathcal{R}$ to represent axis-aligned rectangles in $\mathbb{R}^d$, has received much recent focus [18]. By reducing the Gaussian kernel discrepancy to this problem, led to the best previous results for these ε-KDE coresets [21]. To achieve their result, Matousek *et al.* [18] use a balancing technique of Banaszcyk [3] on a matrix version of discrepancy, by studying the so-call γ_2-norm.

Roughly speaking, we are able to show how to directly reduce the kernel discrepancy problem to the γ_2-norm, and the bound derived from Banaszcyk's Theorem [3] – bypassing the reduction to Tusnady's problem. In particular, the positive definiteness of a kernel, allows us to define a specific gram matrix G which has a real-valued decomposition, which matches the structure studied with the γ_2 norm. Hence, while our positive definite restriction is similar to the characteristic restriction studied for ε-KDE coresets in many other settings [13, 2] it uses a very different aspect of this property: the decomposability, not the embedding.

The lower bound is an extension of a recent lower bound [21] of $\Omega(1/\varepsilon^2)$. The key insight here is that we can reduce the construction which relied on a single $1/\varepsilon^2$-dimensional simplex to a set of $1/(\sqrt{d}\varepsilon)$ d-dimensional simplices. Through some careful analysis, we show applying a modified variant of the argument from the previous lower bound to a particular simplex yields the desired result. Notably, this is the first lower bound for this problem that depends on the dimension d, and this dependence on d matches our upper bound.

2 Upper bound for KDE coreset

Consider a point set $P \subset \mathbb{R}^d$ as input, but as Section 4 describes, it is possible to apply these arguments to other domains. We assume that P is finite and of size n; however, as mentioned in the related work, for many settings, we can reduce this to a point set of size independent of n (size $1/\varepsilon^2$ or d/ε^2, depending on the kernel). Indeed these techniques may start with inputs as continuous distributions as long as we can draw random samples.

To prove our ε-kernel coreset upper bound we introduce two properties that the kernel must have.

- We say a kernel K has c_K*-bounded influence* if, for any $x \in \mathbb{R}^d$ and $\delta > 0$, $|K(x, y)| < \delta$ for all $y \notin x + [-(1/\delta)^{c_K}, (1/\delta)^{c_K}]^d$ for some constant c_K. By default we set $\delta = 1/n$. If c_K is an absolute constant we simply say K is *bounded influence*.
- We say a kernel K is C_k*-Lipschitz* if, for any $x, y, z \in \mathbb{R}^d$, $|K(x, z) - K(y, z)| < C_K \|x - y\|$ for some C_K. If C_K is an absolute constant within the context of the problem, we often just say the kernel is *Lipschitz*.

Next define a lattice $R = \left\{ (\frac{i_1}{\sqrt{d}n}, \frac{i_2}{\sqrt{d}n}, \ldots, \frac{i_d}{\sqrt{d}n}) \mid i_j \text{ are integers} \right\}$. Also, denote, for each $p \in P$, $S_p = p + R \cap [-n^{c_K}, n^{c_K}]^d$ and $S = \cup_{p \in P} S_p$.

The following lemma explains that we only need to consider the evaluation at a finite set (specifically S) rather than the entire space while preserving the discrepancy asymptotically. The advantage of doing this is we can then use the matrix representation of the formula.

▶ **Lemma 1.** $\max_{x \in \mathbb{R}^d} \mathsf{disc}(P, \chi, x) \leq \max_{x \in S} \mathsf{disc}(P, \chi, x) + O(1)$

Proof. For any $x \in \mathbb{R}^d$, if $x \notin \cup_{p \in P} \left(p + [-n^{c_K}, n^{c_K}]^d\right)$, that is x is not within n^{c_K} in all coordinates of some $p \in P$, then $K(p, x) \leq 1/n$ for all $p \in P$. Hence we have

$$\mathsf{disc}(P, \chi, x) = \left| \sum_{p \in P} \chi(p) K(p, x) \right| \leq O(1).$$

Otherwise, pick $x_0 \in S$ be the closest point to x. We have

$$\begin{aligned}
\mathsf{disc}(P,\chi,x) &= \left|\sum_{p\in P}\chi(p)K(p,x)\right| \\
&= \left|\sum_{p\in P}\chi(p)(K(p,x_0)+K(p,x)-K(p,x_0))\right| \\
&\le \left|\sum_{p\in P}\chi(p)K(p,x_0)\right| + \sum_{p\in P}|K(p,x)-K(p,x_0)| \\
&\le \mathsf{disc}(P,\chi,x_0) + \sum_{p\in P} C_K\cdot\|x-x_0\| \\
&\le \mathsf{disc}(P,\chi,x_0) + n\cdot C_K\cdot\sqrt{d(\frac{1}{\sqrt{d}n})^2} \\
&= \mathsf{disc}(P,\chi,x_0) + O(1).
\end{aligned}$$

◀

Now we discuss the matrix view of discrepancy, known results, and then how to map the discretized kernel discrepancy problem into this setting. Consider any $s \times t$ matrix A, and define

$$\mathsf{disc}(A) = \min_{x\in\{-1,+1\}^t} \|Ax\|_\infty .$$

Following Matousek *et al.* [18] we define $\gamma_2(A) = \min_{BC=A} l_1 \cdot l_2$ where l_1 is largest Euclidean norm of row vectors of B and l_2 is largest Euclidean norm of column vectors of C. There is an equivalent geometric interpretation of γ_2. Let $\mathcal{E}_A$ be the set of ellipsoids in $\mathbb{R}^s$ that contain all column vectors of A. Then, $\gamma_2(A) = \min_{E\in\mathcal{E}_A}\max_{x\in E}\|x\|_\infty$. It is easy to see that γ_2 is a norm and $\gamma_2(A) \le \gamma_2(A')$ when the columns of A are subset of the columns of A'. We will apply these properties shortly.

A recent result by Matousek *et al.* [18] shows the following property about connecting discrepancy to γ_2, which was recently made constructive in polynomial time [4].

▶ **Lemma 2** (Matousek *et al.* [18]). *For an $s \times t$ matrix A, $\mathsf{disc}(A) \le O(\sqrt{\log s})\cdot\gamma_2(A)$.*

Let the size of S be $m = O(n^{O(d)})$, and define an $m \times n$ matrix G so its rows are indexed by $x \in S$ and columns indexed by $p \in P$, and $G_{x,p} = K(p,x)$. By examination, $\mathsf{disc}(G) = \min_\chi \max_{x\in S}\mathsf{disc}(P,\chi,x)$.

▶ **Lemma 3.** $\gamma_2(G) = 1$.

Proof. Denote G' be a $m \times m$ matrix with both row and column indexed $x, y \in S$ such that $G'_{x,y} = K(x,y)$. Note that columns of G are a subset of columns of G' since $P \subset S$. Since K is positive definite kernel, it means that G' can be expressed as H^TH for some matrix H. Now denote v_x as the xth column of H for all $x \in S$. We have $v_x^T v_x = G'_{x,x} = 1$ which means the norm $\|v_x\| = \sqrt{v_x^T v_x}$ of each column $v_x \in H$ is 1. Hence the same holds for rows in H^T, and this bounds $\gamma_2(G') \le 1$. Then since $\gamma_2(G) \le \gamma_2(G')$ we have $\gamma_2(G) \le 1$.

On the other hand, one of the coordinates in a column of G is 1. By the geometric definition, any ellipsoid containing columns of G has a point inside of it such that one of its coordinate is 1. Hence $\gamma_2(G) \ge 1$. ◀

Combining all above lemmas, for any $P \subset \mathbb{R}^d$ of size n

$$\begin{aligned}
\mathsf{disc}(n, \mathcal{K}) &\leq \max_{P:|P|=n} \min_{\chi} \max_{x \in S} \mathsf{disc}(P, \chi, x) + O(1) && \text{Lemma 1} \\
&= \max_{P:|P|=n} \mathsf{disc}(G) + O(1) && \text{Definition of } G \\
&\leq O(\sqrt{d \log n} \cdot \gamma_2(G)) && \text{Lemma 2 [18]} \\
&= O(\sqrt{d \log n}). && \text{Lemma 3}
\end{aligned}$$

▶ **Theorem 4.** *Let $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a bounded influence, Lipschitz, positive definite kernel. For any integer n, $\mathsf{disc}(n, \mathcal{K}_d) = O(\sqrt{d \log n})$.*

▶ **Corollary 5.** *Let $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a bounded influence, Lipschitz, positive definite kernel. For any set $P \subset \mathbb{R}^d$, there is a subset $Q \subset P$ of size $O(\frac{1}{\varepsilon}\sqrt{d \log \frac{1}{\varepsilon}})$ such that*

$$\max_{x \in \mathbb{R}^d} |\mathrm{KDE}_P(x) - \mathrm{KDE}_Q(x)| < \varepsilon.$$

Proof. In order to apply the standard halving technique [8, 19], we need to make sure the coloring has the property that half of point assigned $+1$ and the other half of them assigned -1. We adapt a standard idea from combinatorial discrepancy [17].

This can be done by adding an all-one row to the discrepancy matrix G. It guarantees that the difference of number of $+1$ and -1 is $O(\sqrt{d \log n})$ since γ_2 is a norm and therefore we can apply the triangle inequality. Namely,

$$\gamma_2\left(\begin{bmatrix} \mathbb{1}_{1\times n} \\ G \end{bmatrix}\right) \leq \gamma_2\left(\begin{bmatrix} O_{1\times n} \\ G \end{bmatrix}\right) + \gamma_2\left(\begin{bmatrix} \mathbb{1}_{1\times n} \\ O_{m\times n} \end{bmatrix}\right)$$

where $\mathbb{1}$ is all-one matrix and O is zero matrix. Suppose there are more $+1$ than -1. Choose $O(\sqrt{d \log n})$ points assigned $+1$ arbitrarily and flip them to -1 such that it makes the difference zero. Let $P_+ = \{p \in P \mid \chi(p) = +1\}$ and $P_- = \{p \in P \mid \chi(p) = -1\}$. Also, P'_+ and P'_- are defined in the same way after flipping the value. For any $x \in \mathbb{R}^d$,

$$\begin{aligned}
&\left|\sum_{p \in P'_+} K(x,p) - \sum_{p \in P'_-} K(x,p)\right| \\
&\leq \left|\sum_{p \in P_+} K(x,p) - \sum_{p \in P_-} K(x,p)\right| + \left|\sum_{p \in P'_+ \setminus P_+} K(x,p)\right| + \left|\sum_{p \in P_- \setminus P'_-} K(x,p)\right| \\
&= O(\sqrt{d \log n}).
\end{aligned}$$

Now, we can apply the standard halving technique to achieve

$$\max_{x \in \mathbb{R}^d} |\mathrm{KDE}_P(x) - \mathrm{KDE}_Q(x)| < \varepsilon. \qquad \blacktriangleleft$$

Implementation. Note that we do not need to decompose the entire matrix G. Instead, we just need a set of vectors $V = \{v_p \mid p \in P\}$ such that the inner product $\langle v_{p_1}, v_{p_2} \rangle = K(p_1, p_2)$ as input to the algorithm in [4]. This set V can be computed in $\mathsf{poly}(n, d) = \mathsf{poly}(n)$ time assuming $d < n$. Using the standard Merge-Reduce framework [19], the coreset with desired size can be constructed in $O(n\,\mathsf{poly}(1/\varepsilon))$ time.

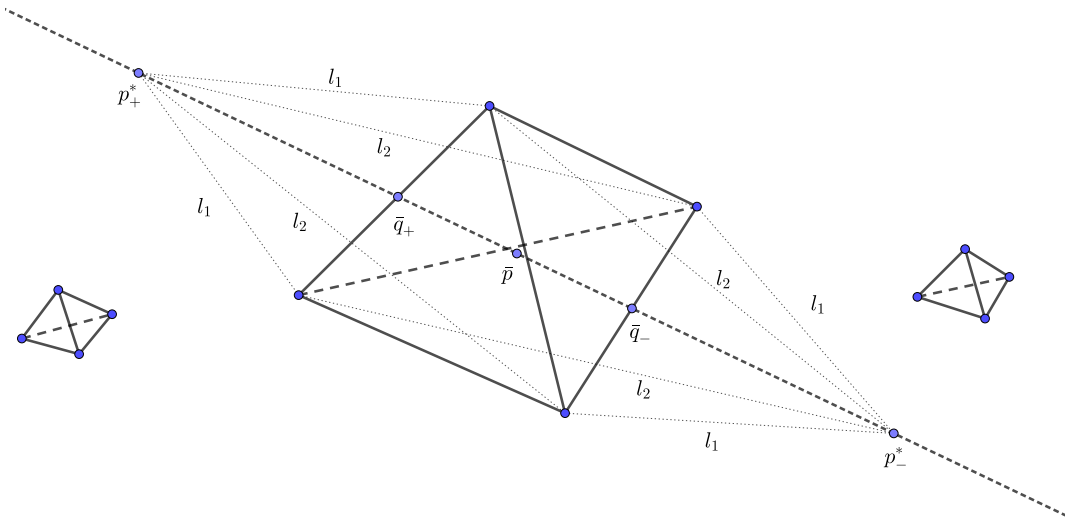


Figure 1 Illustration of the lower bound construction.

3 Lower bound for KDE coreset

In this section, we add two new conditions on our kernel; both of these are common properties of kernels.

- A kernel K is *rotation- and shift-invariant* if there exists a function f such that $K(x, y) = f(\|x - y\|^2)$.
- A rotation- and shift-invariant kernel is *somewhere C_f-steep* if there exist a constant $C_f > 0$, and values $z_f > r_f > 0$ such that $f(z_1) - f(z_2) > C_f \cdot (z_2 - z_1)$ for all $z_1 \in (z_f - r_f, z_f)$ and $z_2 \in (z_f, z_f + r_f)$. When C_f is an absolute constant, we often just say the kernel is *somewhere steep*.

Phillips [20] constructed an example of P of size $1/\varepsilon$ where each point in P is far away from all others. Therefore, if one of them is not picked for a KDE coreset Q, the evaluation of KDE_Q at that point has large error. More recently, Phillips and Tai [21] gave another example of P of size $1/\varepsilon^2$ in an appropriately scaled simplex; that spans $\mathbb{R}^{1/\varepsilon^2}$. They showed that it produces error of $\Omega(1/\sqrt{k})$ at some point, if k is the number of points picked. The following construction combines these two approaches. We divide $n = \frac{\sqrt{d}}{\varepsilon}$ points into n/d groups where each group has d points that form a simplex, and each group is far away from all other groups. It means that there is a group producing $\Omega(1/\sqrt{d})$ error and then, since we have n/d groups, the final error would be $\Omega(\frac{1/\sqrt{d}}{n/d}) = \Omega(\varepsilon)$.

▶ **Theorem 6.** *Consider a rotation- and shift-invariant, somewhere steep, bounded influence kernel K. Assume $d \geq \frac{9z_f^2}{r_f^2}$, where z_f and r_f are absolute constants that depend on K and are defined as they pertain to the somewhere steep criteria. There is a set of $P \in \mathbb{R}^d$ such that, for any subset Q of size $k \leq \frac{\sqrt{d}}{2\varepsilon}$, there is a point $x \in \mathbb{R}^d$ such that $|\text{KDE}_P(x) - \text{KDE}_Q(x)| > \varepsilon$.*

Proof. Let $n = \sqrt{d}/\varepsilon$. We allow weighted coresets of Q; that is, for each $q \in Q$, there is a real number β_q such that $\text{KDE}_Q(x) = \sum_{q \in Q} \beta_q K(x, q)$.

Let $k \leq n/2$ be the size of the potential coreset we consider. Construct P with size of n in $\mathbb{R}^d$ as follow. Let $\{e_i\}_{i=1}^d$ is the standard basis and L is a very large number. Set

$P_j = \left\{p_{i,j} = \sqrt{\frac{z_f}{2}}e_i + jLe_1 \mid i = 1, 2, \ldots, d\right\}$ for all $j = 1, 2, \ldots, \frac{n}{d}$. Define $P = \cup_{j=1}^{n/d} P_j$. Namely, we divide n points into $\frac{n}{d}$ groups and each group has d points which forms a d-simplex. Also, the groups are sufficiently far away from each other. Suppose $Q = \cup_{j=1}^{n/d} \{p_{i_a,j} \mid a = 1, 2, \ldots, k_j\}$ where k_j is the number of points in Q at group j. Denote $Q_j = \{p_{i_a,j} \mid a = 1, 2, \ldots, k_j\}$. That is, $Q = \cup_{j=1}^{n/d} Q_j$ and $|Q_j| = k_j \leq d$ with $\sum_{j=1}^{n/d} k_j = |Q| = k$.

Since $\sum_{j=1}^{n/d} |Q_j| = k \leq n/2$, at least one j must satisfy $k_j \leq \frac{d}{2}$. Denote j' to be that j. We can assume $k_{j'} = d/2$, otherwise, pick enough points arbitrarily from $P_j \setminus Q_{j'}$ and place them in $Q_{j'}$ to make $|Q_{j'}| = k_{j'} = d/2$, but set the corresponding weight to be 0. Denote $\bar{p} = \frac{1}{d}\sum_{p \in P_{j'}} p$ the mean of $P_{j'}$; $\bar{q}_+ = \frac{2}{d}\sum_{q \in Q_{j'}} q$ the mean of $Q_{j'}$; and $\bar{q}_- = \frac{2}{d}\sum_{q \in P_{j'} \setminus Q_{j'}} q$ the mean of points in $P_{j'}$ not selected into $Q_{j'}$; see Figure 1. Also, denote $p_+^* = \bar{q}_+ + \sqrt{\frac{z_f}{2}}\frac{\bar{q}_+ - \bar{p}}{\|\bar{q}_+ - \bar{p}\|}$ and $p_-^* = \bar{q}_- + \sqrt{\frac{z_f}{2}}\frac{\bar{q}_- - \bar{p}}{\|\bar{q}_- - \bar{p}\|}$; translates of these points away from the mean $\bar{p}$ by a specific vector. Note that $\left\|p_+^* - q\right\|$ is the same for all $q \in Q_{j'}$, denoted by l_1 and $\left\|p_+^* - q\right\|$ is same for all $q \in P_{j'} \setminus Q_{j'}$, denoted by l_2. By symmetry, we also have that $l_1 = \left\|p_-^* - q\right\|$ for all $q \in P_{j'} \setminus Q_{j'}$ and $l_2 = \left\|p_-^* - q\right\|$ for all $q \in Q_{j'}$.

If $\sum_{q \in Q_{j'}} \beta_q \geq d/n$, we evaluate the error at p_+^*.

$$
\begin{aligned}
&(\mathrm{KDE}_Q - \mathrm{KDE}_P)(p_+^*) \\
&\quad = \sum_{q \in Q_{j'}} (\beta_q - \frac{1}{n}) f(\left\|p_+^* - q\right\|^2) + \sum_{q \in P_{j'} \setminus Q_{j'}} (-\frac{1}{n}) f(\left\|p_+^* - q\right\|^2) + s \\
&\quad \geq \frac{d}{2n}(f(l_1^2) - f(l_2^2)) + s
\end{aligned}
$$

where $|s|$ is arbitrarily small due to the choice of arbitrarily large number L and the fact that K is bounded influence. If $\sum_{q \in Q_{j'}} \beta_q \leq d/n$, we evaluate the error at p_-^*.

$$
\begin{aligned}
&(\mathrm{KDE}_P - \mathrm{KDE}_Q)(p_-^*) \\
&\quad = \sum_{q \in P_{j'} \setminus Q_{j'}} \frac{1}{n} f(\left\|p_-^* - q\right\|^2) + \sum_{q \in Q_{j'}} (\frac{1}{n} - \beta_q) f(\left\|p_-^* - q\right\|^2) + s \\
&\quad \geq \frac{d}{2n}(f(l_1^2) - f(l_2^2)) + s
\end{aligned}
$$

Therefore, in either case, we need to bound $f(l_1^2) - f(l_2^2)$ from below.

By direct computation, we have $l_1^2 = z_f - \frac{z_f}{d}$ and $l_2^2 = z_f + \frac{z_f}{d} + \frac{2z_f}{\sqrt{d}}$. By enforcing that

$$z_f - r_f < z_f - \frac{z_f}{d} = l_1^2 < z_f$$

and

$$z_f < z_f + \frac{z_f}{d} + \frac{2z_f}{\sqrt{d}} = l_2^2 < z_f + \frac{3z_f}{\sqrt{d}} < z_f + r_f,$$

we can invoke the somewhere C_f-steep property that there exists an x in $\mathbb{R}^d$ for which the inequality holds. Therefore,

$$f(l_1^2) - f(l_2^2) > C_f \cdot (l_2^2 - l_1^2) > C_f \cdot z_f \cdot \frac{2}{\sqrt{d}}.$$

Hence, the error is at least

$$\frac{d}{2n}(f(l_1^2) - f(l_2^2)) + s > \frac{d}{2n}\left(C_f \cdot z_f \cdot \frac{2}{\sqrt{d}}\right) + s > \frac{\sqrt{d}}{n} \cdot C_f \cdot z_f + s = \Omega(\sqrt{d}/n) = \Omega(\varepsilon).\blacktriangleleft$$

4 Applications to specific kernels

In this section, we work through the straight-forward application of these bounds to some specific kernels and settings.

Gaussian and Laplace kernels. These kernels are defined over $\mathbb{R}^d$. They have bounded influence, so $|K(x,p)| \leq \frac{1}{n}$ for all $p \notin [-n^{c_K}, n^{c_K}]^d$ for $c_K = 1$. They are also C_K-Lipschitz with constant $C_K = \alpha$, so $|K(x,z) - K(p,z)| \leq C_K \|x - p\|$ for any $x, p \in \mathbb{R}^d$. These properties imply we can invoke the discrepancy upper bound in Theorem 4.

These kernels are also rotation- and shift-invariant, and somewhere steep with constant $C_f = (\alpha/2)\exp(-\alpha^2)$. Hence we can invoke the lower bound in Theorem 6.

▶ **Corollary 7.** *For Gaussian or Laplacian kernels, for any set $P \in \mathbb{R}^d$, there is a ε-KDE coreset of size $O((\sqrt{d}/\varepsilon)\sqrt{\log 1/\varepsilon})$, and it cannot have an ε-KDE coreset of size $o(\sqrt{d}/\varepsilon)$.*

The Gaussian kernel has an amazing decomposition property that in $\mathbb{R}^d$ if we fix any d' coordinates in any way, then conditioned on those, the remaining $d - d'$ coordinates still follow a Gaussian distribution. Among other things, this means it is useful to construct kernels for complex scenarios. For instance, consider a large set T of n trajectories, each with k waypoints; e.g., backpacking or road trips or military excursions with k nights, and let the waypoints be the (x,y)-coordinates for the location of each night stay. We can measure the similarity between two trajectories $t = (p_1, p_2, \ldots, p_k)$ and $t' = (p'_1, p'_2, \ldots, p'_k)$ as the average similarity between the corresponding waypoints, and we can measure the similarity of any two corresponding waypoints p_j and p'_j with a 2-dimensional Gaussian. Then, by the decomposition property, the full similarity between the trajectories is precisely a $(2k)$-dimensional Gaussian. We can thus define a kernel density estimate over these trajectories KDE_T using this $(2k)$-dimensional Gaussian kernel. Now, given Corollary 7 we know that to approximate KDE_T with a much smaller data set $S \subset T$ so $\|\mathrm{KDE}_T - \mathrm{KDE}_S\|_\infty \leq \varepsilon$, we can construct S so $|S| = O(\sqrt{k}/\varepsilon \cdot \sqrt{\log 1/\varepsilon})$ but cannot in general achieve $|S| = o(\sqrt{k}/\varepsilon)$.

Jensen-Shannon and Hellinger kernels. In order to apply our technique on Δ^d, observe that Δ^d is a subset of a $(d-1)$-dimensional Euclidian subspace of $\mathbb{R}^d$; so we can simply create the grid needed for Lemma 1 within this subspace. Recall that these two kernel have the form of $\exp(-\alpha \mathtt{d}(x,y))$ where $\mathtt{d}(x,y) = \mathtt{d}_{\mathsf{JS}}(x,y) = H(\frac{x+y}{2}) - \frac{H(x)+H(y)}{2}$ for Jensen-Shannon kernel and $\mathtt{d}(x,y) = \mathtt{d}_{\mathsf{H}}(x,y) = \sum_{i=1}^d (\sqrt{x_i} - \sqrt{y_i})^2$ for Hellinger and note that $|K(x,z) - K(y,z)| \leq \alpha\,|\mathtt{d}(x,z) - \mathtt{d}(y,z)|$ for any $x, y, z \in \Delta^d$. It is easy to estimate that when x, y are sufficiently close, for JS kernel, $|\mathtt{d}(x,z) - \mathtt{d}(y,z)| \leq 2d \max_i |x_i - y_i|\,|\log |x_i - y_i||} \leq 2d \max_i \sqrt{|x_i - y_i|}$ and for Hellinger kernel, $|\mathtt{d}(x,z) - \mathtt{d}(y,z)| \leq 4d \max_i \sqrt{|x_i - y_i|}$. So even though these kernels are not Lipschitz, we can still modify the construction of the grid in Lemma 1 with width $\frac{1}{n^4}$ (assuming $d \leq n$) instead of $\frac{1}{\sqrt{d}n}$ such that if x, y lie in the same cell then $|K(x,z) - K(y,z)| = O(\frac{1}{n})$ for any $x, y, z \in \Delta^d$. Since all relevant points are in a bounded domain both kernels have c_K-bounded influence; setting $c_K = 1$ is sufficient.

▶ **Corollary 8.** *For Jensen-Shannon and Hellinger kernels, for any set $P \in \Delta^d$, there is a ε-KDE coreset of size $O((\sqrt{d}/\varepsilon)\sqrt{\log 1/\varepsilon})$.*

Note that these kernels are not rotation- and shift-invariant and therefore our lower bound result does not apply.

These kernels are based on widely-used information distances: the Jensen-Shannon distance $\mathtt{d}_{\mathsf{JS}}(x,p)$ and the Hellinger distance $\mathtt{d}_{\mathsf{H}}(x,p)$. These make sense when the input data

$x, p \in \Delta^d$ represent a "histogram," a discrete probability distribution over a d-variate domain. These are widely studied objects in information theory, and more commonly text analysis. For instance, a common text modeling approach is to represent each document v in a large corpus of documents V (e.g., a collection of tweets, or news articles, or wikipedia pages) as a set of word counts. That is, each coordinate v_j of v represents the number of times that word (indexed by) j occurs in that document. To remove length information from the documents (retaining only the topics), it is common to normalize each vector as $v \mapsto \frac{v}{\|v\|}$ so the jth coordinate represents the probability that a random word on the page is j. The most common modeling choice to measure distance between these distribution representations of documents are the Hellinger and Jensen-Shannon distances, and hence the most natural choice of similarity are the corresponding kernels we examine. In particular, with a very large corpus V of size n, Corollary 8 shows that we can approximate KDE_V, a kernel density estimate of V, with one described by a much smaller set $S \subset V$ so $\| \text{KDE}_V - \text{KDE}_S \| \leq \varepsilon$ and so $|S| = O(\sqrt{d}/\varepsilon \cdot \sqrt{\log 1/\varepsilon})$. Noteably, when one has a fairly large d, and desires high accuracy (small ε), then our new result will provide the best possible ε-KDE coreset.

Exponential kernels. In order to apply our technique on $\mathbb{S}^d$, we can rewrite the kernel to be $K'(x, y) = K(\frac{x}{\|x\|}, \frac{y}{\|y\|})$ for all $x, y \in \mathbb{R}^d \backslash \{0\}$. We construct the grid in Lemma 1 on $\mathbb{R}^d$ for K' and then only retain grid points which lie in the annulus $\mathbb{A}^d = \left\{x \in \mathbb{R}^d \mid \frac{1}{2} \leq \|x\| \leq \frac{3}{2}\right\}$. This annulus contains all grid points which could be the closest point of some point on $\mathbb{S}^d$, as required in Lemma 1. Moreover K' is C_K-Lipschitz on the annulus: it satisfies for any $x, y, z \in \mathbb{A}^d$ that $|K'(x, z) - K'(y, z)| \leq C_K \|x - y\|$, with $C_K = 4\alpha$. Since the domain is restricted to $\mathbb{S}^d$, similar to on the domain Δ^d, any kernel has c_K-bounded influence and setting $c_K = 1$ is sufficient.

▶ **Corollary 9.** *For the exponential kernel, for any set $P \in \mathbb{S}^d$, there is a ε-KDE coreset of size $O((\sqrt{d}/\varepsilon)\sqrt{\log 1/\varepsilon})$.*

The exponential kernel is not rotation- and shift-invariant and therefore our lower bound result does not apply.

Sinc kernel. Note that the sinc kernel is not everywhere positive, and as a result of its structure the VC-dimension is unbounded, so the approaches requiring those properties [16, 20] cannot be applied. It is also not characteristic, so the embedding-based results [13, 2] do not apply either. As a result, there is no non-trivial ε-KDE coreset for the sinc kernel. However, in our approach, the positivity of one single entry in the discrepancy matrix does not matter so long as the entire matrix is positive definite – which is the case for sinc. Therefore, our result could be applied to sinc kernel, with $c_K = 1$ (it has 1-bounded influence), $C_K = \alpha/\pi$ (it is (α/π)-Lipschitz) and $C_f = \alpha^2/2\pi^2$ (it is somewhere $(\alpha^2/2\pi^2)$-steep).

▶ **Corollary 10.** *For sinc kernels, for any set $P \in \mathbb{R}^d$, there is a ε-KDE coreset of size $O((1/\varepsilon)\sqrt{\log 1/\varepsilon})$ (for $d = \{1, 2, 3\}$), and it cannot have a ε-KDE coreset of size $\Omega(1/\varepsilon)$.*

5 Conclusion

We proved that Gaussian kernel has a ε-KDE coreset of size $O(\frac{1}{\varepsilon}\sqrt{d \log \frac{1}{\varepsilon}})$ and the size must satisfy $\Omega(\frac{\sqrt{d}}{\varepsilon})$; both upper and lower bound result can be extended to a broad class of kernels. In particular the upper bound only requires that the kernel be positive definite (typically the same restriction needed for most machine learning techniques) and that it has a domain

which can be discretized over a bounded region without inducing too much error. This family of applicable kernels includes new options like the sinc kernel, which while positive definite in $\mathbb{R}^d$ for $d = \{1, 2, 3\}$, it is not characteristic, is not always positive, and its super-level sets do not have bounded VC-dimension. This is the first non-trivial ε-KDE coreset result for these kernels.

By inspecting the new constructive algorithm for obtaining small discrepancy in the γ_2-norm [4], the extra $\sqrt{\log}$ factor comes from the union bound over the randomness in the algorithm. Indeed, a previous result [21] showed that if $d = \frac{1}{\varepsilon^2}$ then the upper bound is $O(\frac{1}{\varepsilon^2})$, which is tight. This bound is deterministic and does not have an extra $\sqrt{\log}$ factor. Therefore, a natural conjecture is that the upper bound result can be further improved to $O(\sqrt{d}/\varepsilon)$, at least in a well-behaved setting like for the Gaussian kernel.

There are many other, even more diverse kernels which are positive definite, which operate on domains as diverse as graphs, time series, strings, and trees [15]. The heart of the upper bound construction which uses the decomposition of the associated positive definite matrix will work even for these kernels. However, it is less clear how to generate a finite gram or discrepancy matrix G, whose size depends polynomially on the data sets size for these discrete objects. Such constructions would further expand the pervasiveness of the ε-KDE coreset technique we present.

References

1 Ery Arias-Castro, David Mason, and Bruno Pelletier. On the estimation of the gradient lines of a density and the consistency of the mean-shift algorithm. *Journal of Machine Learning Research*, 17(43):1–28, 2016.

2 Francis Bach, Simon Lacoste-Julien, and Guillaume Obozinski. On the equivalence between herding and conditional gradient algorithms. In *ICML 2012 International Conference on Machine Learning*, 2012.

3 Wojciech Banaszczyk. Balancing vectors and gaussian measures of n-dimensional convex bodies. *Random Structures & Algorithms*, 12(4):351–360, 1998.

4 Nikhil Bansal, Daniel Dadush, Shashwat Garg, and Shachar Lovett. The Gram-Schmidt walk: A cure for the Banaszczyk blues (to appear). *Proceedings of the fiftieth annual ACM symposium on Theory of computing*, 2018.

5 Jon Louis Bentley and James B. Saxe. Decomposable searching problems I: Static-to-dynamic transformations. *Journal of Algorithms*, 1(4), 1980.

6 Omer Bobrowski, Sayan Mukherjee, and Jonathan E. Taylor. Topological consistency via kernel estimation. *Bernoulli*, 23:288–328, 2017.

7 Bernard Chazelle. *The Discrepancy Method*. Cambridge, 2000.

8 Bernard Chazelle and Jiri Matousek. On linear-time deterministic algorithms for optimization problems in fixed dimensions. *J. Algorithms*, 21:579–597, 1996.

9 Luc Devroye and László Györfi. *Nonparametric Density Estimation: The L_1 View*. Wiley, 1984.

10 Petros Drineas and Michael W. Mahoney. On the Nyström method for approximating a Gram matrix for improved kernel-based learning. *JLMR*, 6:2153–2175, 2005.

11 Jianqing Fan and Irene Gijbels. *Local polynomial modelling and its applications: monographs on statistics and applied probability 66*, volume 66. CRC Press, 1996.

12 Brittany Terese Fasy, Fabrizio Lecci, Alessandro Rinaldo, Larry Wasserman, Sivaraman Balakrishnan, and Aarti Singh. Confidence sets for persistence diagrams. *The Annals of Statistics*, 42:2301–2339, 2014.

13 Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Scholkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13:723–773, 2012.

14 Matthias Hein and Olivier Bousquet. Hilbertian metrics and positive definite kernels on probability measures. In *AISTATS*, pages 136–143, 2005.

15 Thomas Hofmann, Bernhard Schölkopf, and Alexander J. Smola. A review of kernel methods in machine learning. Technical Report 156, Max Planck Institute for Biological Cybernetics, 2006.

16 Sarang Joshi, Raj Varma Kommaraji, Jeff M Phillips, and Suresh Venkatasubramanian. Comparing distributions and shapes using the kernel distance. In *Proceedings of the twenty-seventh annual symposium on Computational geometry*, pages 47–56. ACM, 2011.

17 Jiri Matousek. *Geometric Discrepancy; An Illustrated Guide, 2nd printing*. Springer-Verlag, 2010.

18 Jiri Matousek, Aleksandar Nikolov, and Kunal Talwar. Factorization norms and hereditary discrepancy. *arXiv preprint arXiv:1408.1376*, 2014.

19 Jeff M. Phillips. Algorithms for ε-approximations of terrains. In *ICALP*, 2008.

20 Jeff M Phillips. ε-samples for kernels. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 1622–1632. SIAM, 2013.

21 Jeff M Phillips and Wai Ming Tai. Improved coresets for kernel density estimates. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2718–2727. SIAM, 2018.

22 Jeff M. Phillips, Bei Wang, and Yan Zheng. Geometric inference on kernel density estimates. In *SOCG*, 2015.

23 Alessandro Rinaldo and Larry Wasserman. Generalized density clustering. *The Annals of Statistics*, pages 2678–2722, 2010.

24 Isaac J Schoenberg. Metric spaces and completely monotone functions. *Annals of Mathematics*, pages 811–841, 1938.

25 Bernhard Scholkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.

26 Erich Schubert, Arthur Zimek, and Hans-Peter Kriegel. Generalized outlier detection with flexible kernel density estimates. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pages 542–550. SIAM, 2014.

27 David W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization*. Wiley, 1992.

28 Bernard W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall/CRC, 1986.

29 Bharath K. Sriperumbudur, Arthur Gretton, Kenji Fukumizu, Bernhard Schölkopf, and Gert R. G. Lanckriet. Hilbert space embeddings and metrics on probability measures. *JMLR*, 11:1517–1561, 2010.

30 Yan Zheng and Jeff M. Phillips. l_∞ error and bandwidth selection for kernel density estimates of large data. In *KDD*, 2015.

Optimal Analysis of an Online Algorithm for the Bipartite Matching Problem on a Line

Sharath Raghvendra
Virginia Tech
Blacksburg, USA
sharathr@vt.edu

Abstract

In the online metric bipartite matching problem, we are given a set S of server locations in a metric space. Requests arrive one at a time, and on its arrival, we need to immediately and irrevocably match it to a server at a cost which is equal to the distance between these locations. A α-competitive algorithm will assign requests to servers so that the total cost is at most α times the cost of M_{OPT} where M_{OPT} is the minimum cost matching between S and R.

We consider this problem in the adversarial model for the case where S and R are points on a line and $|S| = |R| = n$. We improve the analysis of the deterministic Robust Matching Algorithm (RM-Algorithm, Nayyar and Raghvendra FOCS'17) from $O(\log^2 n)$ to an optimal $\Theta(\log n)$. Previously, only a randomized algorithm under a weaker oblivious adversary achieved a competitive ratio of $O(\log n)$ (Gupta and Lewi, ICALP'12). The well-known Work Function Algorithm (WFA) has a competitive ratio of $O(n)$ and $\Omega(\log n)$ for this problem. Therefore, WFA cannot achieve an asymptotically better competitive ratio than the RM-Algorithm.

2012 ACM Subject Classification Theory of computation → Online algorithms

Keywords and phrases Bipartite Matching, Online Algorithms, Adversarial Model, Line Metric

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.67

Related Version A full version of this paper is available at `https://arxiv.org/abs/1803.07206`.

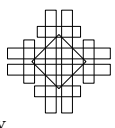
Funding This work is supported by a NSF CRII grant NSF-CCF 1464276

1 Introduction

Driven by consumers' demand for quick access to goods and services, business ventures schedule their delivery in real-time, often without the complete knowledge of the future request locations or their order of arrival. Due to this lack of complete information, decisions made tend to be sub-optimal. Therefore, there is a need for competitive *online algorithms* which immediately and irrevocably allocate resources to requests in real-time by incurring a small cost.

Motivated by these real-time delivery problems, we study the problem of computing the online metric bipartite matching of requests to servers. Consider servers placed in a metric space where each server has a capacity that restricts how many requests it can serve. When a new request arrives, one of the servers with positive capacity is matched to this request. After this request is served, the capacity of the server reduces by one. We assume that the cost associated with this assignment is a metric cost; for instance, it could be the minimum distance traveled by the server to reach the request.

The case where the capacity of every server is ∞ is the celebrated k-server problem. The case where every server has a capacity of 1 is the *online metric bipartite matching problem.*




In this case, the requests arrive one at a time, we have to immediately and irrevocably match it to some unmatched server. We assume there are n servers and n requests. The resulting assignment is a matching and is referred to as an *online matching*. An optimal assignment is impossible since an adversary can easily fill up the remaining locations of requests in R in a way that our current assignment becomes sub-optimal. Therefore, we want our algorithm to compute an assignment that is competitive with respect to the optimal matching. For any input S, R and any arrival order of requests in R, we say our algorithm is α-*competitive*, for $\alpha > 1$, when the cost of the online matching M is at most α times the minimum cost, i.e.,

$$w(M) \leq \alpha w(M_{\text{OPT}}).$$

Here M_{OPT} is the minimum-cost matching of the locations in S and R. In the above discussion, note the role of the adversary. In the *adversarial model*, the adversary knows the server locations and the assignments made by the algorithm and generates a sequence to maximize α. In the weaker oblivious adversary model, the adversary knows the randomized algorithm but does not know the random choices made by the algorithm. In this paper, we consider the online metric bipartite matching problem in the adversarial model and where S and R are points on a line.

Consider the adversarial model. For any request, the *greedy heuristic* simply assigns the closest unmatched server to it. The greedy heuristic, even for the line metric, is only 2^n-competitive [3] for the online matching problem. The well-known Work Function Algorithm (WFA) chooses a server that minimizes the sum of the *greedy cost* and the so-called *retrospective cost*. For the k-server problem, the competitive ratio of the WFA is $2k - 1$ [6] which is near optimal with a lower bound of k on the competitive ratio of any algorithm in any metric space that has at least $k + 1$ points [7].

In the context of online metric matching problem, there are algorithms that achieve a competitive ratio of $2n - 1$ in the adversarial model [9, 5, 4]. This competitive ratio is worst-case optimal, i.e., there exists a metric space where we cannot do better. However, for Euclidean metric, for a long time, there was a stark contrast between the upper bound of $2n - 1$ and the lower bound of $\Omega(n^{1-1/d})$. Consequently, significant effort has been made to study the performance of online algorithms in special metric spaces, especially the line metric. For example, for the line metric, it has been shown that the WFA when applied to the online matching problem has a lower bound of $\Omega(\log n)$ and an upper bound of $O(n)$ [2]; see also [1] for a $O(n^{0.585})$competitive algorithm for the line metric. In the oblivious adversary model, there is an $O(\log n)$-competitive [3] algorithm for the line metric. There is also an $O(d \log n)$-competitive algorithm in the oblivious adversary model for any metric space with doubling dimension d [3].

Only recently, for any metric space and for the adversarial model, Raghvendra and Nayyar [8] provided a bound of $O(\mu(S) \log^2 n)$ on the competitive ratio of the RM-Algorithm – an algorithm that was introduced by Raghvendra [9]; here $\mu(S)$ is the worst case ratio of the the TSP and diameter of any positive diameter subset of S. There is a simple lower bound of $\Omega(\mu(S))$ on the competitive ratio of any algorithm for this problem. Therefore, RM-Algorithm is near-optimal for every input set S. When S is a set of points on a line, $\mu(S) = 2$ and therefore, their analysis bounds the competitive ratio of the RM-Algorithm by $O(\log^2 n)$ for the line metric and $O(n^{1-1/d} \log^2 n)$ for any d-dimensional Euclidean metric. Furthermore, RM-Algorithm also has a lower bound of $\Omega(\log n)$ on its competitive ratio for the line metric. In this paper, we provide a different analysis and show that the RM-Algorithm is $\Theta(\log n)$-competitive.

Overview of RM-Algorithm. At a high-level, the RM-Algorithm maintains two matchings M and M^*, both of which match requests seen so far to the same subset of servers in S. We refer to M as the online matching and M^* as the offline matching. For a parameter $t > 1$ chosen by the algorithm, the offline matching M^* is a t-approximate minimum-cost matching satisfying a set of relaxed dual feasibility conditions of the linear program for the minimum-cost matching; here each constraint relaxed by a multiplicative factor t. Note that, when $t = 1$, the matching M^* is simply the minimum-cost matching.

When the i^{th} request r_i arrives, the algorithm computes an augmenting path P_i with the minimum "cost" for an appropriately defined cost function. This path P_i starts at r_i and ends at an unmatched server s_i. The algorithm then augments the offline matching M^* along P whereas the online matching M will simply match the two endpoints of P_i. Note that M and M^* will always match requests to the same subset of servers. We refer to the steps taken by the algorithm to process request r_i as the i^{th} phase of the algorithm. For $t > 1$, it has been shown in [9] that the sum of the costs of every augmenting path computed by the algorithm is bounds the online matching cost from above. Nayyar and Raghvendra [8] use this property and bounded the ratio of the sum of the costs of augmenting paths to the cost of the optimal matching. In order to accomplish this they associate every request r_i to the cost of P_i. To bound the sum of costs of all the augmenting paths, they partition the requests into $\Theta(\log^2 n)$ groups and within each group they bound this ratio by $\mu(S)$ (which is a constant when S is a set of points on a line). For the line metric, each group can, in the worst-case, have a ratio of $\Theta(1)$. However, not all groups can simultaneously exhibit this worst-case behavior. In order to improve the competitive ratio from $O(\log^2 n)$ to $O(\log n)$, therefore, one has to bound the combined ratios of several groups by a constant making the analysis challenging.

1.1 Our results and techniques

In this paper, we show that when the points in $S \cup R$ are on a line, the RM-Algorithm achieves a competitive ratio of $O(\log n)$. Our analysis is tight as there is an example in the line metric for which the RM-Algorithm produces an online matching which is $\Theta(\log n)$ times the optimal cost. We achieve this improvement using the following new ideas:

- First, we establish the the *ANFS-property* of the RM-Algorithm (Section 3.1). We show that many requests are matched to an approximately nearest free server (ANFS) by the algorithm. We define certain edges of the online matching M as *short* edges and show that every short edge matches the request to an approximately nearest free server. Let M_H be the set of short edges of the online matching and $M_L = M \setminus M_H$ be the *long* edges. We also show that when $t = 3$, the total cost of the short edges $w(M_H) \geq w(M)/6$ and therefore, the cost of the long edges is $w(M_L) < (5/6)w(M)$.
- For every point in $S \cup R$, the RM-Algorithm maintains a dual weight (Section 2). For our analysis in the line metric, we assign an interval to every request. The length of this interval is determined by the dual weight of the request. At the start of phase i, let σ_{i-1} be the union of all such intervals. By its construction, σ_{i-1} will consist of a set of interior-disjoint intervals. While processing request r_i, the RM-Algorithm conducts a series of dual adjustments and a subset of requests (referred to as B_i) undergo an increase in dual weights. After the dual adjustments, the union of the intervals for requests in B_i forms a single interval and these increases conclude in the discovery of the minimum cost augmenting path. Therefore, after phase i, intervals in σ_{i-1} may grow and combine to form a single interval $\mathcal{I}$ in σ_i. Furthermore, the new online edge (s_i, r_i) is also contained inside this newly formed interval $\mathcal{I}$ (Section 3.3). Based on the length of the interval $\mathcal{I}$, we assign one of $O(\log n)$ levels to the edge (s_i, r_i). This partitions all the online edges in $O(\log n)$ levels.

- The online edges of any given level k can be expressed as several non-overlapping well-aligned matching of a well separated input (Section 4) . We establish properties of such matchings (Section 3.2 and Figure 1) and use it to bound the total cost of the "short" online edges of level k by the sum of $w(M_{\text{OPT}})$ and γ times the cost of the long edges of level k, where γ is a small positive constant (Section 4). Adding across the $O(\log n)$ levels, we get $w(M_H) \leq O(\log n)w(M_{\text{OPT}}) + \gamma w(M_L)$. Using the ANFS-property of short and long edges, we immediately get $(1/6 - 5\gamma/6)w(M) \leq O(\log n)w(M_{\text{OPT}})$. For a sufficiently small γ, we bound the competitive ratio, i.e., $w(M)/w(M_{\text{OPT}})$ by $O(\log n)$.

Organization. The rest of the paper is organized as follows. We begin by presenting (in Section 2) the RM-Algorithm and some of its use properties as shown in [9]. For the analysis, we establish the *ANFS-property* in Section 3.1. After that, we will (in Section 3.2) introduce well aligned matchings of well-separated inputs on a line. Then, in Section 3.3, we interpret the dual weight maintained for each request as an interval and study the properties of the union of these intervals. Using these properties (in Section 4) along with the ANFS-property of the algorithm, we will establish a bound of $O(\log n)$ on the competitive ratio for the line metric.

2 Background and algorithm details

In this section, we introduce the relevant background and describe the RM-algorithm.

A *matching* $M \subseteq S \times R$ is any set of vertex-disjoint edges of the complete bipartite graph denoted by $G(S, R)$. The cost of any edge $(s, r) \in S \times R$ is given by $d(s, r)$; we assume that the cost function satisfies the metric property. The cost of any matching M is given by the sum of the costs of its edges, i.e., $w(M) = \sum_{(s,r)\in M} d(s, r)$. A *perfect matching* is a matching where every server in S is serving exactly one request in R, i.e., $|M| = n$. A *minimum-cost perfect matching* is a perfect matching with the smallest cost.

Given a matching M^*, an *alternating path* (resp. cycle) is a simple path (resp. cycle) whose edges alternate between those in M^* and those not in M^*. We refer to any vertex that is not matched in M^* as a *free vertex*. An *augmenting path* P is an alternating path between two free vertices. We can *augment* M^* by one edge along P if we remove the edges of $P \cap M^*$ from M^* and add the edges of $P \setminus M^*$ to M^*. After augmenting, the new matching is precisely given by $M^* \oplus P$, where $\oplus$ is the symmetric difference operator. A matching M^* and a set of dual weights, denoted by $y(v)$ for each point $v \in S \cup R$, is a *t-feasible matching* if, for any request $r \in R$ and a server $s \in S$, the following conditions hold:

$$y(s) + y(r) \leq td(s, r), \tag{1}$$

$$y(s) + y(r) = d(s, r) \quad \text{if } (s, r) \in M^*. \tag{2}$$

Also, we refer to an edge $(s, r) \in S \times R$ to be *eligible* if either $(s, r) \in M^*$ or (s, r) satisfies inequality (1) with equality:

$$y(s) + y(r) = td(s, r), \quad \text{if } (s, r) \notin M^* \tag{3}$$

$$y(s) + y(r) = d(s, r) \quad \text{if } (s, r) \in M^*. \tag{4}$$

For a parameter $t \geq 1$, we define the *t-net-cost* of any augmenting path P with respect to M^* to be:

$$\phi_t(P) = t \left(\sum_{(s,r)\in P\setminus M^*} d(s, r) \right) - \sum_{(s,r)\in P\cap M^*} d(s, r).$$

The definitions of t-feasible matching, eligible edges and t-net cost (when $t = 1$) are also used in describing the well-known Hungarian algorithm which computes the minimum-cost matching. In the Hungarian algorithm, initially $M^* = \emptyset$ is a 1-feasible matching with all the dual weights $y(v)$ set to 0. In each iteration, the Hungarian Search procedure adjusts the dual weights $y(\cdot)$ and computes an augmenting path P of eligible edges while maintaining the 1-feasibility of M^* and then augments M^* along P. The augmenting path P computed by the standard implementation of the Hungarian search procedure can be shown to also have the minimum 1-net-cost.

Using this background, we describe the RM-Algorithm. At the start, the value of t is chosen at the start of the algorithm. The algorithm maintains two matchings: an online matching M and a t-feasible matching (also called the *offline matching*) M^* both of which are initialized to $\emptyset$. After processing $i-1$ requests, both matchings M and M^* match each of the $i-1$ requests to the same subset of servers in S, i.e., the set of free (unmatched) servers S_F is the same for both M and M^*. To process the i^{th} request r_i, the algorithm does the following

1. Compute the minimum t-net-cost augmenting path P_i with respect to the offline matching M^*. Let P_i be this path starting from r_i and ending at some server $s_i \in S_F$.
2. Update offline matching M^* by augmenting it along P_i, i.e., $M^* \leftarrow M^* \oplus P_i$ and update online matching M by matching r_i to s_i. $M \leftarrow M \cup \{(s_i, r_i)\}$.

While computing the minimum t-net-cost augmenting path in Step 1, if there are multiple paths with the same t-net-cost, the algorithm will simply select the one with the fewest number of edges. Throughout this paper, we set $t = 3$. In [9], we present an $O(n^2)$-time search algorithm that is similar to the Hungarian Search to compute the minimum t-net-cost path in Step 1 any phase i and we describe this next.

The implementation of Step 1 of the algorithm is similar in style to the Hungarian Search procedure. To compute the minimum t-net-cost path P_i , the algorithm grows an alternating tree consisting only of eligible edges. There is an alternating path of eligible edges from r_i to every server and request participating in this tree. To grow this tree, the algorithm increases the dual weights of every request in this tree until at least one more edge becomes eligible and a new vertex enters the tree. In order to maintain feasibility, the algorithm reduces the dual weights of all the servers in this tree by the same amount. This search procedure ends when an augmenting path P_i consisting only of eligible edges is found. Let B_i (resp. A_i) be the set of requests (resp. servers) that participated in the alternating tree of phase i. Note that during Step 1, the dual weights of requests in B_i may only increase and the dual weights of servers in A_i may only reduce.

The second step begins once the augmenting path P_i is found. The algorithm augments the offline matching M^* along this path. Note that, for the M^* to be t-feasible, the edges that newly enter M^* must satisfy (2). In order to ensure this, the algorithm will reduce the dual weight of each request r on P_i to $y(r) \leftarrow y(r) - (t-1)d(s,r)$. Further details of the algorithm and proof of its correctness can be found in [9]. In addition, it has also been shown that the algorithm maintains the following three invariants:

(I1) The offline matching M^* and dual weights $y(\cdot)$ form a t-feasible matching,

(I2) For every server $s \in S$, $y(s) \leq 0$ and if $s \in S_F$, $y(s) = 0$. For every request $r \in R$, $y(r) \geq 0$ and if r has not yet arrived, $y(r) = 0$,

(I3) At the end of the first step of phase i of the algorithm the augmenting path P_i is found and the dual weight of r_i, $y(r_i)$, is equal to the t-net-cost $\phi_t(P_i)$.

Notations. Throughout the rest of this paper, we will use the following notations. We will index the requests in their order of arrival, i.e., r_i is the ith request to arrive. Let R_i be the set of first i request. Our algorithm processes the request r_i, by computing an augmenting path P_i. Let s_i be the free server at the other end of the augmenting path P_i. Let $\mathbb{P} = \{P_1, \ldots, P_n\}$ be the set of n augmenting paths generated by the algorithm. In order to compute the augmenting path P_i, in the first step, the algorithm adjusts the dual weights and constructs an alternating tree; let B_i be the set of requests and let A_i be the set of servers that participate in this alternating tree. Let M_i^* be the offline matching after the ith request has been processed; i.e., the matching obtained after augmenting the matching M_{i-1}^* along P_i. Note that $M_0^* = \emptyset$ and $M_n^* = M^*$ is the final matching after all the n requests have been processed. The online matching M_i is the online matching after i requests have been processed. M_i consists of edges $\bigcup_{j=1}^{i}(s_j, r_j)$. Let S_F^i be the free servers with respect to matchings M_{i-1} and M_{i-1}^*, i.e., the set of free servers at the start of phase i. For any path P, let $\ell(P) = \sum_{(s,r)\in P} d(s,r)$ be its *length*.

Next, in Section 3.1, we will present the approximate nearest free server (ANFS) property of the RM-Algorithm. In Section 3.2, we present an well aligned matching of a well separated input instance. In Section 3.3, we interpret the execution of each phase of the RM-Algorithm in the line metric. Finally, in Section 4, we give our analysis of the algorithm for the line metric.

3 New properties of the algorithm

In this section, we present new properties of the RM-Algorithm. First, we show that the RM-Algorithm will assign an approximate nearest free server for many requests and show that the total cost of these "short" matches will be at least one-sixth of the online matching cost. Our proof of this property is valid for any metric space.

3.1 Approximate nearest free server property

We divide the n augmenting paths $\mathbb{P} = \{P_1, \ldots, P_n\}$ computed by the RM-Algorithm into two sets, namely *short* and *long* paths. For any i, we refer to P_i as a *short* augmenting path if $\ell(P_i) \leq \frac{4}{t-1}\phi_t(P_i)$ and *long* otherwise. Let $H \subseteq \mathbb{P}$ be this set of all short augmenting paths and $L = \mathbb{P} \setminus H$ be the *long* augmenting paths. In phase i, the algorithm adds an edge between s_i and r_i in the online matching. We refer to any edge of the online matching (s_i, r_i) as a *short edge* if P_i is a short augmenting path. Otherwise, we refer to this edge as a *long* edge. The set of all short edges, M_H and the set of long edges M_L partition the edges of the online matching M.

At the start of phase i, S_F^i are the set of free servers. Let $s^* \in S_F^i$ be the server closest to r_i, i.e., $s^* = \arg\min_{s\in S_F^i} d(r_i, s)$. Any other server $s \in S_F^i$ is an *α-approximate nearest free server* to the request r if

$$d(r_i, s) \leq \alpha d(r_i, s^*).$$

In Lemma 1 and 2, we show that the short edges in M_H match a request to a 6-ANFS and the cost of $w(M_H)$ is at least one-sixth of the cost of the online matching.

▶ **Lemma 1.** *For any request r_i, if P_i is a short augmenting path, then s_i is a $(4+\frac{4}{t-1})$-ANFS of r_i.*

Proof. Let s^* be the nearest available server of r_i in S_F^i. Both s^* and r_i are free and so the edge $P = (s^*, r_i)$ is also an augmenting path with respect to M_{i-1}^* with $\phi_t(P) = td(s^*, r_i)$.

The algorithm computes P_i which is the minimum t-net-cost path with respect to M^*_{i-1} and so,

$$\phi_t(P_i) \leq td(s^*, r_i).$$

Since P_i is a short augmenting path,

$$\frac{(t-1)}{4} d(s_i, r_i) \leq \phi_t(P_i) \leq td(s^*, r_i), \tag{5}$$

$$d(s_i, r_i) \leq \frac{4t}{t-1} d(s^*, r_i) = (4 + \frac{4}{t-1}) d(s^*, r_i), \tag{6}$$

implying that s_i is a $(4 + \frac{4}{t-1})$-approximate nearest free server to the request r_i. ◀

▶ **Lemma 2.** *Let M_H be the set of short edges of the online matching M. Then,*

$$(4 + \frac{4}{t-1}) w(M_H) \geq w(M). \tag{7}$$

If we set $t = 3$, then $6w(M_H) \geq w(M)$ or $w(M_H) \geq w(M)/6$.

Convention and notations for the line metric. When S and R are points on a line, any point $v \in S \cup R$ is simply a real number. We can interpret any edge $(s, r) \in S \times R$ as the line segment that connects s and r and its cost $d(s, r)$ as $|s - r|$. We will abuse notation and use (s, r) to denote both the edge as well as the corresponding line segment. A matching of S and R, therefore, is also a set of line segments, one corresponding to each edge. For any closed interval $I = [x, y]$, $open(I)$ will be the open interval (x, y). We define the boundary, $bd(I)$ of any closed (or open) interval $I = [x, y]$ (or $I = (x, y)$) to be the set of two end points$\{x, y\}$. Any edge (s, r) of the matching is called a *left edge* if the server s is to the left of request r. Otherwise, we refer to the edge as the *right edge*.

3.2 Properties of 1-dimensional matching

In this section, we define certain input instances that we refer to as a well-separated input instance. We then define matchings that are well-aligned for such instance and then bound the cost of such a well-aligned matching by the cost of their optimal matching. In Section 4, we divide the edges of the online matching into well-aligned matchings on well-separated instances. This will play a critical role in bounding the competitive ratio for the line metric.

Well-separated instances and well aligned matchings. A *well-separated instance* of the 1-dimensional matching problem is defined as follows. For any $\Delta > 0$ and $0 < \varepsilon \leq 1/8$, consider the following four intervals $I_M = [0, \Delta]$, $I_L = [-\varepsilon\Delta, 0], I_R = [\Delta, (1 + \varepsilon)\Delta]$ and $I_A = [-\varepsilon\Delta, (1 + \varepsilon)\Delta]$. Note that I_L is the leftmost interval, I_R is the rightmost interval, and I_M lies in the middle of the I_L and I_R. I_A is simply the union of I_L, I_R, and I_M. We say that any input set of servers S and requests R is an ε-well-separated input if, for some $\Delta > 0$, there is a translation of $S \cup R$ such that $S \subset I_L \cup I_R$ and $R \subset I_A$.

Given an ε-well-separated input S and R, consider the intervals, $I'_L = [-\varepsilon\Delta, \varepsilon\Delta]$ and $I'_R = [(1 - \varepsilon)\Delta, (1 + \varepsilon)\Delta]$. We divide the edges of any matching M of S and R into three groups. Any edge $(s, r) \in M$ is *close* if $s, r \in I'_L$ or $s, r \in I'_R$. Any edge $(s, r) \in M$ is a *far* edge if $(s, r) \in I'_L \times I'_R$ or $(s, r) \in I'_R \times I'_L$. Let M_{close} and M_{far} denote the close and far edges of M respectively. For a matching edge (s, r), we denote it as a *medium edge* if the request r is inside the interval $[\varepsilon\Delta, (1 - \varepsilon\Delta)]$ and the server s is inside the interval I_L or

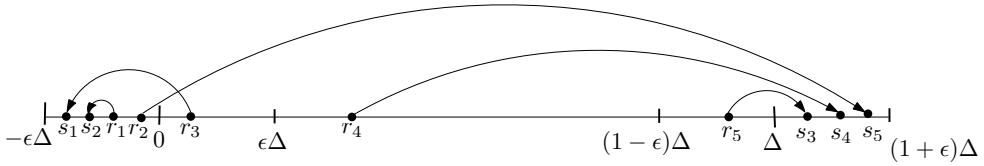


Figure 1 The server set is $S = \{s_1, s_2, s_3, s_4, s_5\} \subset [-\varepsilon\Delta, 0] \cup [\Delta, (1+\varepsilon)\Delta]$ and request set is $R = \{r_1, r_2, r_3, r_4, r_5\} \subset [-\varepsilon\Delta, (1+\varepsilon)\Delta]$. So, $S \cup R$ is an ε-well separated input instance. The matching $M = \{(r_1, s_2), (r_2, s_5), (r_3, s_1), (r_4, s_4), (r_5, s_3)\}$ is partitioned into $M_{\text{close}} = \{(r_1, s_2), (r_3, s_1), (r_5, s_3)\}$, $M_{\text{far}} = \{(r_2, s_5)\}$ and $M_{\text{med}} = \{(r_4, s_4)\}$. M is ε-well aligned since the edges of M_{close} in $[-\varepsilon\Delta, (1+\varepsilon)\Delta]$ are left edges (server is to the left of request) and those in $[(1-\varepsilon)\Delta, (1+\varepsilon)\Delta]$ are right edges (server is to the right of request).

I_R. We denote this set of edges as the *medium edges* and denote it by M_{med}. From the well-separated property of the input it is easy to see that $M = M_{\text{far}} \cup M_{\text{med}} \cup M_{\text{close}}$. A matching M is *ε-well-aligned* if all the edges of M_{close} with both their endpoints inside I'_R (resp. I'_L) are right (resp. left) edges. See Figure 1 for an example of ε-well aligned matching of an ε-well separated input instance. Any ε-well-aligned matching of an ε-well-separated input instance satisfies the following property.

▶ **Lemma 3.** *For any $0 \le \varepsilon \le 1/8$, given an ε-well-separated input S and R and an ε-well-aligned matching M, let M_{OPT} be the optimal matching of S and R and let $M_{\text{close}}, M_{\text{far}}$ and M_{med} be as defined above. Then,*

$$w(M_{\text{close}}) + w(M_{\text{med}}) \le \left(\frac{2}{\varepsilon} + 3\right) w(M_{\text{OPT}}) + \frac{4\varepsilon}{1-2\varepsilon} w(M_{\text{far}}).$$

3.3 Interpreting dual weights for the line metric

Next, we interpret the dual weights and their changes during the RM-Algorithm for the line metric and derive some of its useful properties.

Span of a request. For any request $r \in R$, let $y^i_{max}(r)$ be the largest dual weight that is assigned to r in the first i phases. The second step of phase i does not increase the dual weights of requests, and so, $y^i_{max}(r)$ must be a dual weight assigned at the end of first step of some phase $j \le i$. For any request r and any phase i, the span of r, denoted by $span(r, i)$, is an open interval that is centered at r and has a length of $\frac{2y^i_{max}(r)}{t}$, i.e., $span(r, i) = \left(r - \frac{y^i_{max}(r)}{t}, r + \frac{y^i_{max}(r)}{t}\right)$. We will refer to the closed interval $[r - \frac{y^i_{max}(r)}{t}, r + \frac{y^i_{max}(r)}{t}]$ with center r and length $\frac{2y^i_{max}(r)}{t}$ as $cspan(r, i)$.

Intuitively, request r may participate in one or more alternating trees in the first i phases of the algorithm. The dual weight of every request that participates in an alternating tree, including r, increases. These increases reflect their combined search for a free server. The region $span(r, i)$ represents the region swept by r in its search for a free server in the first i phases. We show in Lemma 4 that the span of any request does not contain a free server in its interior. We show that, during the search, if the span of a request r expands to include a free server $s \in S_F$ on its boundary, i.e., $s \in bd(span(r, i))$, then the algorithm would have found a minimum t-net-cost path and the search would stop. Therefore, the open interval $span(r, i)$ will remain empty.

▶ **Lemma 4.** *For every r, $span(r, i) \cap S^i_F = \emptyset$.*

Proof. For the sake of contradiction, we assume that the span of a request r contains a free server s, i.e., $s \in S_F^i \cap span(r,i)$. So, the distance between r and s is $|s-r| < \frac{y^i_{max}(r)}{t}$, or,

$$y^i_{max}(r) > t(|s-r|). \tag{8}$$

Since $y^i_{max}(r)$ is the largest dual weight assigned to r, there is a phase $j \leq i$, when request r is assigned this dual weight by the algorithm. Since the first step of the algorithm may only increase and the second step may only decrease the dual weights of any request, we can assume that $y(r)$ was assigned the dual weight of $y^i_{max}(r)$ at the end of the first step of some phase j. Let $y(s)$ and $y(r) = y^i_{max}(r)$ be the dual weights at the end of the first step of phase j. From Invariant (I1), it follows that $y^i_{max}(r) + y(s) \leq t(|s-r|)$. From this and (8), we have $y(s) < 0$. The free server s has $y(s) < 0$ contradicting invariant (I2). ◀

▶ **Lemma 5.** *Let (s,r) be any eligible edge at the end of the first step of phase i. Then,*

$$y^i_{max}(r) \geq t|s-r|,$$

implying that $s \in cspan(r,i)$ and the edge $(s,r) \subset span(r,i)$.

Proof. An edge is eligible if it is in M^*_{i-1} or if it satisfies (3). Suppose $(s,r) \notin M^*_{i-1}$ and satisfies (3). In this case, $y(s)+y(r) = t(|s-r|)$. From (I2), $y(s) \leq 0$ and so, $y(r) \geq t(|s-r|)$, implying that $y^i_{max}(r) \geq t(|s-r|)$.

For the case where $(s,r) \in M^*_{i-1}$, let $0 < j < i$ be the largest index such that $(s,r) \in M^*_j$ and $(s,r) \notin M^*_{j-1}$. Therefore, $(s,r) \in P_j \setminus M^*_{j-1}$. Since P_j is an augmenting path with respect to M^*_{j-1}, every edge of $P_j \setminus M^*_{j-1}$ satisfies the eligibility condition (4) at the end of the first step of phase j of the algorithm. For any vertex v, let $y'(v)$ be its dual weight after the end of the first step of phase j of the algorithm. From (4), we have $y'(r)+y'(s) \leq t|s-r|$. From (I2), since $y'(s) \leq 0$, we have $y'(r) \geq t|s-r|$. By definition, $y^i_{max}(r) \geq y'(r)$ and therefore $y^i_{max}(r) \geq t|s-r|$. ◀

Search interval of a request. Recollect that B_i is the set of requests that participate in the alternating tree of phase i. In Lemma 6, we show that $\bigcup_{r \in B_i} cspan(r,i)$ is a single closed interval. We define *search interval* of r_i, denoted by $sr(r_i)$, as the open interval $open(\bigcup_{r \in B_i} cspan(r,i))$. The search interval of a request represents the region searched for a free server by all the requests of B_i. In Lemma 6, we establish a useful property of search interval of a request. We show that the search interval of r_i does not contain any free servers of S^i_F and the free server s_i (the match of r_i in the online matching) is at the boundary of $sr(r_i)$. Since the search interval contains r_i, it follows that s_i is either the closest free server to the left or the closest free server to the right of r_i. Using the fact that all requests of B_i are connected to r_i by an path of eligible edges along with Lemma 4 and Lemma 5, we get the proof for Lemma 6.

▶ **Lemma 6.** *After phase i, $\bigcup_{r \in B_i} cspan(r,i)$ is a single closed interval and so, the search interval $sr(r_i)$ of any request r_i is a single open interval. Furthermore,*

- *All edges between A_i and B_i are inside the search interval of r_i, i.e.,$A_i \times B_i \subseteq sr(r_i)$,*
- *There are no free servers inside the search interval of r_i, i.e.,$S^i_F \cap sr(r_i) = \emptyset$, and,*
- *The server s_i chosen by the algorithm is on the boundary of the search interval of r_i, i.e., $s_i \in bd(sr(r_i))$.*

Cumulative search region. After phase i, the *cumulative search region* csr_i for the first i requests R_i is the union of the individual search intervals $\text{csr}_i = (sr(r_1) \cup sr(r_2) \ldots \cup sr(r_i))$. Since the cumulative search region is the union of open intervals, it is formed by a set of pairwise interior-disjoint open intervals. Let σ_i of csr_i be a sequence of these interior-disjoint open intervals in the cumulative search region ordered from left to right, i.e., $\sigma_i = \langle \mathcal{C}^i_1, \ldots, \mathcal{C}^i_k \rangle$, where $\mathcal{C}^i_j$ appears before $\mathcal{C}^i_l$ in the sequence if and only if the interval $\mathcal{C}^i_j$ is to the left of interval $\mathcal{C}^i_l$. In Lemma 7, we establish properties of the cumulative search region. We show that every edge of the online matching M_i and the offline matching M^*_i is contained inside some interval of the sequence σ_i. We also show that there are no free servers inside any interval of σ_i, i.e., $S^i_F \cap \text{csr}_i = \emptyset$.

▶ **Lemma 7.** *After phase i of the algorithm,*

- *For every edge $(s, r) \in M^*_i \cup M_i$ there exists an interval $\mathcal{C} \in \sigma_i$ such that $(s, r) \subseteq \mathcal{C}$,*
- *The set of free servers S^i_F satisfies $S^i_F \cap csr_i = \emptyset$, and there is an interval $\mathcal{C}$ in σ_i such that the server $s_i \in bd(\mathcal{C})$.*

When a new request r_{i+1} is processed by the algorithm, the search interval $sr(r_{i+1})$ is added to the cumulative search region, i.e., $\text{csr}_{i+1} = \text{csr}_i \cup sr(r_{i+1})$. Suppose $\langle \mathcal{C}^i_j, \ldots, \mathcal{C}^i_l \rangle$ intersects $sr(r_{i+1})$, then csr_{i+1} will have a single interval that contains intervals $\langle \mathcal{C}^i_j, \ldots, \mathcal{C}^i_l \rangle$ and $sr(r_{i+1})$. From this description, an easy observation follows:

(O1) Given two intervals $\mathcal{C}$ and $\mathcal{C}'$ in cumulative search regions σ_i and σ_j respectively, with $j > i$, then either $\mathcal{C} \cap \mathcal{C}' = \emptyset$ or $\mathcal{C} \subseteq \mathcal{C}'$.

Matchings in cumulative search regions. From the property of cumulative search region established in Lemma 7, every edge of the online matching M_i and the offline matching M^*_i is contained inside some interval of the sequence σ_i. We denote the edges of online and offline matching that appear in an interval $\mathcal{C}$ of σ_i by $M_{\mathcal{C}}$ and $M^*_{\mathcal{C}}$ respectively. Therefore, $M_i = \bigcup_{\mathcal{C} \in \sigma_i} M_{\mathcal{C}}$ and $M^*_i = \bigcup_{\mathcal{C} \in \sigma_i} M^*_{\mathcal{C}}$. Note that $M_{\mathcal{C}}$ and $M^*_{\mathcal{C}}$ match the same set of servers and requests. Let this set of servers and requests be denoted by $S_{\mathcal{C}}$ and $R_{\mathcal{C}}$ respectively.

Consider any sequence of interior-disjoint intervals $\langle \mathcal{C}^{i_1}_{j_1}, \ldots, \mathcal{C}^{i_k}_{j_k} \rangle$, where each interval $\mathcal{C}^{i_l}_{j_l}$ appears in the cumulative search region σ_{i_l}. Note that every interval in this set can appear in after the execution of a different phase. In Lemma 8, we show that $\sum_{l=1}^{k} w(M^*_{\mathcal{C}^{i_l}_{j_l}}) \leq tw(M_{\text{OPT}})$, i.e., the total cost of the subset of offline matching edges that are contained inside all of these disjoint intervals is within a factor of t times the cost of the optimal matching. This property, therefore, relates the cost of subsets of offline matchings, each of which appear at different phases to the optimal cost.

▶ **Lemma 8.** *For any i, j, let $\mathcal{C}^i_j$ be the jth interval in the sequence σ_i. Suppose we are given intervals $\mathcal{C}^{i_1}_{j_1}, \mathcal{C}^{i_2}_{j_2} \ldots, \mathcal{C}^{i_k}_{j_k}$ such that no two of these intervals have any mutual intersection. Then $w(M^*_{\mathcal{C}^{i_1}_{j_1}}) + w(M^*_{\mathcal{C}^{i_2}_{j_2}}) + \ldots + w(M^*_{\mathcal{C}^{i_k}_{j_k}}) \leq tw(M_{\text{OPT}})$.*

4 Analysis for the line metric

For each interval of any cumulative search region σ_i, we assign it a level between 0 and $O(t \log(nt))$ based on the length of the interval. We also partition the edges of the online matching into $O(t \log(nt))$ levels.

Level of an interval. For any $0 < i \leq n$, we assign a *level* to every interval of any cumulative search region σ_i. The *level* of any interval $\mathcal{C}_j^i \in \sigma_i$, denoted by $lev(\mathcal{C}_j^i)$, is k if

$$(1 + \frac{1}{32t})^k (w(M_{\text{OPT}})/n) \leq \mathcal{L}(\mathcal{C}_j^i) \leq (1 + \frac{1}{32t})^{k+1} (w(M_{\text{OPT}})/n).$$

Here $\mathcal{L}(\mathcal{C}_j^i)$ is the length of the interval $\mathcal{C}_j^i$. All intervals whose length $\mathcal{L}(\mathcal{C}_j^i) \leq w(M_{\text{OPT}})/n$ is assigned a level 0.

Level of an online edge. For any request r_i and its match s_i in M_i, let r_i and s_i be contained in an interval $\mathcal{C}' \in \sigma_i$. Then, the level of this edge (r_i, s_i), denoted by $lev(r_i)$, is given by the level of the interval $\mathcal{C}'$, i.e., $lev(r_i) = lev(\mathcal{C}')$.

▶ **Lemma 9.** *The largest level of assigned to any online edge is* $O(t \log(nt))$.

Tracking the evolution of cumulative search region. The cumulative search region csr_i after any phase i will include disjoint intervals from the sequence σ_i. When we process a new request r_{i+1}, the cumulative search region is updated to include the open interval $sr(r_{i+1})$. This may combine a contiguous sub-sequence of intervals $\Gamma_{i+1} = \langle \mathcal{C}_j^i, \mathcal{C}_{j+1}^i, \ldots, \mathcal{C}_l^i \rangle$ of σ_i along with the interval $sr(r_{i+1})$ to form a single open interval $\mathcal{C}_j^{i+1}$ in σ_{i+1}. We refer to the sequence of intervals Γ_{i+1} as the *predecessors* of $\mathcal{C}_j^{i+1}$ and denote $i+1$ as the *birth phase* of $\mathcal{C}_j^{i+1}$. For each interval $\mathcal{C}$ in Γ_{i+1}, we define its *successor* to be $\mathcal{C}_j^{i+1}$ and denote $i+1$ as the *death phase* of $\mathcal{C}$. Suppose $\mathcal{C}_j^{i+1}$ is a level k interval. Then, it is easy to see that there is at most one level k interval in Γ_{i+1}.

▶ **Lemma 10.** *For* $k \geq 1$ *and any level* k *interval* $\mathcal{C}_j^{i+1}$*, there is at most one level* k *interval in the predecessor set* Γ_{i+1}.

Proof. For the sake of contradiction, suppose there are at least two interior-disjoint level k intervals $\mathcal{C}$ and $\mathcal{C}'$. Since $\mathcal{C}_j^{i+1}$ contains both $\mathcal{C}$ and $\mathcal{C}'$, its length is at least $2(1 + \frac{1}{32t})^k (w(M_{\text{OPT}})/n)$ contradicting the fact that $\mathcal{C}_j^{i+1}$ is a level k interval. ◀

Recollect that $M_{\mathcal{C}_j^{i+1}}$ and $M^*_{\mathcal{C}_j^{i+1}}$ are edges of the online and offline matching inside $\mathcal{C}_j^{i+1}$that match the servers of $S_{\mathcal{C}_j^{i+1}}$ to requests of $R_{\mathcal{C}_j^{i+1}}$. Let $M_{\Gamma_{i+1}} = \bigcup_{\mathcal{C} \in \Gamma_{i+1}} M_{\mathcal{C}}$ be the edges of the online matching contained inside the predecessors Γ_{i+1} of $\mathcal{C}_j^{i+1}$. By construction, the matchings $M_{\mathcal{C}_j^{i+1}}$ and $M_{\Gamma_{i+1}}$ only differ in the edge (s_{i+1}, r_{i+1}). So, $M_{\Gamma_{i+1}}$ match servers in $S_{\mathcal{C}_j^{i+1}} \setminus \{s_{i+1}\}$ to requests in $R_{\mathcal{C}_j^{i+1}} \setminus \{r_{i+1}\}$.

Maximal and minimal level k interval. A level k interval $\mathcal{C}$ is *maximal* if

- $\mathcal{C}$ is an interval in the cumulative search region after all the n requests have been processed, i.e., $\mathcal{C} \in \sigma_n$ or,
- if the successor $\mathcal{C}'$ of $\mathcal{C}$ has $lev(\mathcal{C}) > k$.

A level k interval $\mathcal{C}_j^{i+1}$ is a *minimal level k interval* if none of its predecessors in Γ_{i+1} are of level k.

Next, we will describe a sequence of nested level k intervals that capture the evolution of a minimal level k interval into a maximal level k interval. For any level k interval $\mathcal{C}$, we define a sequence of level k intervals $\mathcal{E}_{\mathcal{C}}$ along with a set $\text{comp}(\mathcal{C})$ of intervals of level strictly less than k in a recursive way as follows: Initially $\mathcal{E}_{\mathcal{C}} = \emptyset$ and $\text{comp}(\mathcal{C}) = \emptyset$. Suppose phase i is the birth phase of $\mathcal{C}$. If all the intervals in the predecessor Γ_i have a level less than

k, then $\mathcal{C}$ is a minimal level k interval. We add $\mathcal{C}$ to to front of the sequence $\mathcal{E}_{\mathcal{C}}$ and add the predecessors of $\mathcal{C}$, Γ_i, to the set comp($\mathcal{C}$) and return $\mathcal{E}_{\mathcal{C}}$ and comp($\mathcal{C}$). Otherwise, from Lemma 10, there is exactly one level k interval $\mathcal{C}' \in \Gamma_i$ and all other intervals have a level strictly less than k. In this case, we compute $\mathcal{E}_{\mathcal{C}'}$ and comp($\mathcal{C}'$) recursively. We set $\mathcal{E}_{\mathcal{C}}$ to the sequence $\mathcal{E}_{\mathcal{C}'}$ followed by $\mathcal{C}$. We add all intervals in $\Gamma_i \setminus \mathcal{C}'$ along with the intervals of comp($\mathcal{C}'$) to comp($\mathcal{C}$) and return comp($\mathcal{C}$) and $\mathcal{E}_{\mathcal{C}}$.

For any maximal level k interval $\mathcal{C}$, consider the sequence $\mathcal{E}_{\mathcal{C}} = \langle \mathcal{C}_1, \ldots, \mathcal{C}_l \rangle$ and the set of disjoint intervals comp($\mathcal{C}$) . From the construction of $\mathcal{E}_{\mathcal{C}}$, the following observations follow in a straight-forward way:

(E1) All intervals $\mathcal{C}' \in \mathcal{E}_{\mathcal{C}}$ are level k intervals. The first and the last interval, i.e., $\mathcal{C}_1$ and $\mathcal{C}_l = \mathcal{C}$ in $\mathcal{E}_{\mathcal{C}}$ are minimal and maximal level k intervals respectively,

(E2) The intervals in $\mathcal{E}_{\mathcal{C}}$ are nested, i.e., $\mathcal{C}_i \subseteq \mathcal{C}_{i+1}$ for all $1 \leq i \leq l-1$.

Let $\mathcal{M}_{\mathcal{C}}$ be the set of all level k edges of the online matching that are contained inside $\mathcal{C}$. Let $\mathcal{S}_{\mathcal{C}}$ and $\mathcal{R}_{\mathcal{C}}$ be the servers and request that are matched by $\mathcal{M}_{\mathcal{C}}$. From the construction of comp($\mathcal{C}$), the following properties follow in a straight-forward way:

(C1) comp($\mathcal{C}$) is a set of disjoint intervals each of which is contained inside $\mathcal{C}$,

(C2) Every point $s \in S_{\mathcal{C}} \setminus \mathcal{S}_{\mathcal{C}}$ and $r \in R_{\mathcal{C}} \setminus \mathcal{S}_{\mathcal{C}}$ is contained inside some interval from the set comp($\mathcal{C}$).

Let $I = \{I_1, \ldots, I_{l_k}\}$ be the set of all maximal level k intervals. In the following lemma, we show that I is a set of pairwise interior-disjoint intervals. Furthermore, the matchings $\mathcal{M}_{I_1}, \mathcal{M}_{I_2}, \ldots, \mathcal{M}_{I_{l_k}}$ partitions all the level k edges of the online matching M.

▶ **Lemma 11.** *Let $I = \{I_1, \ldots, I_{l_k}\}$ be the set of all maximal level k intervals. Then, I is a set of interior-disjoint intervals. Furthermore, the matchings $\mathcal{M}_{I_1}, \mathcal{M}_{I_2}, \ldots, \mathcal{M}_{I_{l_k}}$ partitions all the level k edges of the online matching M.*

For any level k interval $\mathcal{C}$, let $\mathcal{M}_{\mathcal{C}}^{\text{OPT}}$ be the optimal matching of $\mathcal{S}_{\mathcal{C}}$ and $\mathcal{R}_{\mathcal{C}}$. Suppose $\{I_1, \ldots, I_{l_k}\}$ be the set of all maximal level k intervals. Let $M^*_{\text{comp}(I_j)}$ be the union (over all intervals of comp(I_j)), the offline matching contained inside an interval of comp(I_j). By (C2) and Lemma 7, $M^*_{\text{comp}(I_j)}$ matches servers in $S_{I_j} \setminus \mathcal{S}_{I_j}$ to requests $R_{I_j} \setminus \mathcal{R}_{I_j}$. The symmetric difference of $\bigcup_{I_j \in I} M^*_{\text{comp}(I_j)}$and $\bigcup_{I_j \in I} M^*_{I_j}$consists of augmenting paths that match $\bigcup_{I_j \in I} \mathcal{S}_{I_j}$ to $\bigcup_{I_j \in I} \mathcal{R}_{I_j}$. Note that these are precisely the points that are matched by level k online edges. From Lemma 8, $w(\bigcup_{I_j \in I} M^*_{\text{comp}(I_j)}) \leq tw(M_{\text{OPT}})$and $w(\bigcup_{I_j \in I} M^*_{I_j}) \leq tw(M_{\text{OPT}})$ leading to the following lemma.

▶ **Lemma 12.** *For any $k > 0$, let $\{I_1 \ldots, I_{l_k}\}$ be the set of all maximal level k intervals. Then, $\sum_{j=1}^{l_k} w(\mathcal{M}_{I_j}^{\text{OPT}}) \leq 2tw(M_{\text{OPT}})$.*

For $k > 0$, consider any maximal level k interval I_j. In Lemma 13, for $\varepsilon > 1/32t$ and $t = 3$, we show that the matching $\mathcal{M}_{I_j}$ is an ε-well-aligned matching of an ε-well-separated input instance. Additionally, we also show in this lemma that every far edge of this well-aligned matching is also a long edge of the online matching.

▶ **Lemma 13.** *For any level $k \geq 1$ interval $\mathcal{C}$, consider the sequence $\mathcal{E}_{\mathcal{C}}$. Let $\varepsilon = \frac{1}{32t}$ and $t = 3$. Then, $\mathcal{R}_{\mathcal{C}}$ and $\mathcal{S}_{\mathcal{C}}$ is an ε-well separated input and the matching $\mathcal{M}_{\mathcal{C}}$ is an ε-well-aligned matching of $\mathcal{S}_{\mathcal{C}}$ and $\mathcal{R}_{\mathcal{C}}$. Furthermore, each far edge in the ε-well aligned matching $\mathcal{M}_{\mathcal{C}}$ is also a long edge of the online matching.*

Let $\mathcal{M}_{I_j}^{\text{far}}$, $\mathcal{M}_{I_j}^{\text{close}}$ and $\mathcal{M}_{I_j}^{\text{med}}$ denote the far, close and medium edges of $\mathcal{M}_{I_j}$. Recollect that $\mathcal{M}_{I_j}^{\text{far}} \cup \mathcal{M}_{I_j}^{\text{close}} \cup \mathcal{M}_{I_j}^{\text{med}} = \mathcal{M}_{I_j}$. From Lemma 3, we know that

$$w(\mathcal{M}_{I_j}^{\text{close}}) + w(\mathcal{M}_{I_j}^{\text{med}}) \leq (2/\varepsilon + 3) w(\mathcal{M}_{I_j}^{\text{OPT}}) + \frac{4\varepsilon}{1-2\varepsilon} w(\mathcal{M}_{I_j}^{\text{far}}).$$

Adding the above inequality over all level k intervals and setting $\varepsilon = \frac{1}{32t}$ and $t = 3$, we get

$$\sum_{j=1}^{l_k}(w(\mathcal{M}_{I_j}^{\text{close}}) + w(\mathcal{M}_{I_j}^{\text{med}})) \leq (2/\varepsilon + 3)\sum_{j=1}^{l_k} w(\mathcal{M}_{I_j}^{\text{OPT}}) + \frac{4\varepsilon}{1-2\varepsilon}\sum_{j=1}^{l_k} w(\mathcal{M}_{I_j}^{\text{far}}).$$

Adding the above equation for all levels $k > 0$ and adding $w(M_{\text{OPT}})$ to the RHS and LHS, we get

$$w(M_{\text{OPT}}) + \sum_{k=1}^{O(\log n)}\sum_{j=1}^{l_k}(w(\mathcal{M}_{I_j}^{\text{close}}) + w(\mathcal{M}_{I_j}^{\text{med}})) \leq$$

$$w(M_{\text{OPT}}) + (\frac{2}{\varepsilon} + 3)\sum_{j=1}^{l_k} w(\mathcal{M}_{I_j}^{\text{OPT}}) + \frac{2}{47}\sum_{k=1}^{O(\log n)}\sum_{j=1}^{l_k} w(\mathcal{M}_{I_j}^{\text{far}}). \tag{9}$$

Note that every short edge of the online matching with level $k > 0$ will appear, for some maximal level k interval $I_j \in \{I_1, \ldots, I_{l_k}\}$, in $\mathcal{M}_{I_j}^{\text{close}}$ or $\mathcal{M}_{I_j}^{\text{med}}$. By construction, the short edges of level 0 have a length of at most $w(M_{\text{OPT}})/n$ and there are at most n such edges. Recollect that M_H is the set of short online edges. Therefore,

$$w(M_H) \leq w(M_{\text{OPT}}) + \sum_{k=1}^{O(\log n)}\sum_{j=1}^{l_k}(w(\mathcal{M}_{I_j}^{\text{close}}) + w(\mathcal{M}_{I_j}^{\text{med}})).$$

Every edge in $\mathcal{M}_{I_j}^{\text{far}}$ is only a long edge of the online matching. Recollect that M_L is the set of long edges of the online matching. We can, therefore, rewrite the (9) as

$$w(M_H) \leq O(1/\varepsilon)\sum_{k=1}^{O(\log n)}\sum_{j=1}^{l_k} w(\mathcal{M}_{I_j}^{\text{OPT}}) + \frac{2}{47}\, w(M_L),$$

$$w(M_H) \quad \leq \quad O(\log n)w(M_{\text{OPT}}) + \frac{2}{47}w(M_L).$$

The above inequality follows from Lemma 12 and the fact that every online edge appears in exactly one of the maximal level k intervals for some level k. In Lemma 2, by setting $t = 3$, we get $w(M_H) \geq w(M)/6$ and $w(M_L) \leq (5/6)w(M)$, and

$$\begin{aligned} w(M_H) - \frac{2}{47}w(M_L) &\leq O(\log n)w(M_{\text{OPT}}), \\ \frac{w(M)}{6} - \frac{2}{47}\times\frac{5}{6}w(M) &\leq O(\log n)w(M_{\text{OPT}}), \\ \frac{w(M)}{w(M_{\text{OPT}})} &\leq O(\log n). \end{aligned}$$

References

1 Antonios Antoniadis, Neal Barcelo, Michael Nugent, Kirk Pruhs, and Michele Scquizzato. A $o(n)$-competitive deterministic algorithm for online matching on a line. In *Approximation and Online Algorithms: 12th International Workshop, WAOA 2014*, pages 11–22, 2015.

2 Koutsoupias Elias and Nanavati Akash. The online matching problem on a line. In *Approximation and Online Algorithms: First International Workshop, WAOA 2003, Budapest, Hungary, September 16-18, 2003. Revised Papers*, pages 179–191, 2004.

3 A. Gupta and K. Lewi. The online metric matching problem for doubling metrics. In *Proceedings of International Conference on Automata, Languages and Programming*, volume 7391, pages 424–435, 2012.

4 Bala Kalyanasundaram and Kirk Pruhs. Online weighted matching. *J. Algorithms*, 14(3):478–488, 1993.

5 Samir Khuller, Stephen G. Mitchell, and Vijay V. Vazirani. On-line algorithms for weighted bipartite matching and stable marriages. *Theor. Comput. Sci.*, 127(2):255–267, 1994.

6 Elias Koutsoupias and Christos H. Papadimitriou. On the k-server conjecture. *J. ACM*, 42(5):971–983, 1995.

7 Mark S. Manasse, Lyle A. McGeoch, and Daniel D. Sleator. Competitive algorithms for server problems. *J. Algorithms*, 11(2):208–230, 1990.

8 Krati Nayyar and Sharath Raghvendra. An input sensitive online algorithm for the metric bipartite matching problem. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 505–515, 2017.

9 Sharath Raghvendra. A Robust and Optimal Online Algorithm for Minimum Metric Bipartite Matching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2016)*, volume 60, pages 18:1–18:16, 2016.

Almost All String Graphs are Intersection Graphs of Plane Convex Sets

János Pach[1]
Ecole Polytechnique Fédérale de Lausanne and Rényi Institute, Hungarian Academy of Sciences
Lausanne, Switzerland and Budapest, Hungary
pach@cims.nyu.edu

Bruce Reed
School of Computer Science, McGill University, Laboratoire I3S CNRS, and Professor Visitante Especial, IMPA
Montreal, Canada and Rio de Janeiro, Brazil
breed@cs.mcgill.ca

Yelena Yuditsky
School of Computer Science, McGill University
Montreal, Canada
yuditskyL@gmail.com

Abstract

A *string graph* is the intersection graph of a family of continuous arcs in the plane. The intersection graph of a family of plane convex sets is a string graph, but not all string graphs can be obtained in this way. We prove the following structure theorem conjectured by Janson and Uzzell: The vertex set of *almost all* string graphs on n vertices can be partitioned into *five* cliques such that some pair of them is not connected by any edge ($n \to \infty$). We also show that every graph with the above property is an intersection graph of plane convex sets. As a corollary, we obtain that *almost all* string graphs on n vertices are intersection graphs of plane convex sets.

2012 ACM Subject Classification Mathematics of computing → Graph theory, Theory of computation → Randomness, geometry and discrete structures

Keywords and phrases String graph, intersection graph, plane convex set

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.68

Related Version A full version of the paper is available at [PRY18], `http://arxiv.org/abs/1803.06710`.

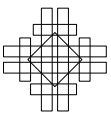
Acknowledgements This research was carried out while all three authors were visiting IMPA in Rio de Janeiro. They would like to thank the institute for its generous support.

1 Overview

The *intersection graph* of a collection C of sets is a graphs whose vertex set is C and in which two sets in C are connected by an edge if and only if they have nonempty intersection. A *curve* is a subset of the plane which is homeomorphic to the interval $[0, 1]$. The intersection graph of a finite collection of curves ("strings") is called a *string graph*.

Ever since Benzer [Be59] introduced the notion in 1959, to explore the topology of genetic structures, string graphs have been intensively studied both for practical applications and

[1] Supported by Swiss National Science Foundation Grants 200021-165977 and 200020-162884.



theoretical interest. In 1966, studying electrical networks realizable by printed circuits, Sinden [Si66] considered the same constructs at Bell Labs. He proved that not every graph is a string graph, and raised the question whether the recognition of string graphs is decidable. The affirmative answer was given by Schaefer and Štefankovič [ScSt04] 38 years later. The difficulty of the problem is illustrated by an elegant construction of Kratochvíl and Matoušek [KrMa91], according to which there exists a string graph on n vertices such that no matter how we realize it by curves, there are two curves that intersect at least 2^{cn} times, for some $c > 0$. On the other hand, it was proved in [ScSt04] that every string graph on n vertices and m edges can be realized by polygonal curves, any pair of which intersect at most $2^{c'm}$ times, for some other constant c'. The problem of recognizing string graphs is NP-complete [Kr91, ScSeSt03].

In spite of the fact that there is a wealth of results for various special classes of string graphs, understanding the structure of general string graphs has remained an elusive task. The aim of this paper is to show that *almost all* string graphs have a very simple structure. That is, the proportion of string graphs that possess this structure tends to 1 as n tends to infinity.

Given any graph property P and any $n \in \mathbb{N}$, we denote by P_n the set of all graphs with property P on the (labeled) vertex set $V_n = \{1, \ldots, n\}$. In particular, $\textsc{String}_n$ is the collection of all string graphs with the vertex set V_n. We say that an n-element set is partitioned into parts of *almost equal size* if the sizes of any two parts differ by at most $n^{1-\epsilon}$ for some $\epsilon > 0$, provided that n is sufficiently large.

▶ **Theorem 1.** *As $n \to \infty$, the vertex set of almost every string graph $G \in \textsc{String}_n$ can be partitioned into 4 parts of almost equal size such that 3 of them induce a clique in G and the 4th one splits into two cliques with no edge running between them.*

▶ **Theorem 2.** *Every graph G whose vertex set can be partitioned into 4 parts such that 3 of them induce a clique in G and the 4th one splits into two cliques with no edge running between them, is a string graph.*

Theorem 1 settles a conjecture of Janson and Uzzell from [JaU17], where a related weaker result was proved in terms of graphons.

We also prove that a typical string graph can be realized using relatively simple strings.

Let $\textsc{Conv}_n$ denote the set of all intersection graphs of families of n labeled convex sets $\{C_1, \ldots, C_n\}$ in the plane. For every pair $\{C_i, C_j\}$, select a point in $C_i \cap C_j$, provided that such a point exists. Replace each convex set C_i by the polygonal curve obtained by connecting all points selected from C_i by segments, in the order of increasing x-coordinate. Observe that any two such curves belonging to different C_is intersect at most $2n$ times. The intersection graph of these curves (strings) is the same as the intersection graph of the original convex sets, showing that $\textsc{Conv}_n \subseteq \textsc{String}_n$. Taking into account the construction of Kratochvíl and Matoušek [KrMa91] mentioned above, it easily follows that the sets $\textsc{Conv}_n$ and $\textsc{String}_n$ are not the same, provided that n is sufficiently large.

▶ **Theorem 3.** *There exist string graphs that cannot be obtained as intersection graphs of convex sets in the plane.*

We call a graph G *canonical* if its vertex set can be partitioned into 4 parts such that 3 of them induce a clique in G and the 4th one splits into two cliques with no edge running between them. The set of canonical graphs on n vertices is denoted by $\textsc{Canon}_n$. Theorem 2 states $\textsc{Canon}_n \subset \textsc{String}_n$. In fact, this is an immediate corollary of $\textsc{Conv}_n \subset \textsc{String}_n$ and the relation $\textsc{Canon}_n \subset \textsc{Conv}_n$, formulated as

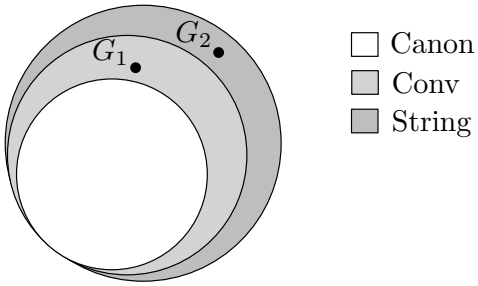


Figure 1 The graph G_1 is the any planar graph with more than 20 vertices. The graph G_2 is the graph from the construction of Kratochvíl and Matoušek [KrMa91].

▶ **Theorem 4.** *The vertices of every canonical graph G can be represented by convex sets in the plane such that their intersection graph is G.*

The converse is not true. Every planar graph can be represented as the intersection graph of convex sets in the plane (Koebe [Ko36]). Since no planar graph contains a clique of size exceeding four, for $n > 20$ no planar graph with n vertices is canonical.

Combining Theorems 1 and 4, we obtain the following.

▶ **Corollary 5.** *Almost all string graphs on n labeled vertices are intersection graphs of convex sets in the plane.*

See Figure 1 for a sketch of the containment relation of the families of graphs discussed above.

The rest of this paper is organized as follows. In Section 2, we recall the necessary tools from extremal graph theory, and adapt a partitioning technique of Alon, Balogh, Bollobás, and Morris [AlBBM11] to analyze string graphs; see Theorem 8. In Section 3, we collect some simple facts about string graphs and intersection graphs of plane convex sets, and combine them to prove Theorem 4. In Section 4, we strengthen Theorem 8 in two different ways and, hence, prove Theorem 1 modulo a small number of exceptional vertices. We wrap up the proof of Theorem 1 in Section 5.

2 The structure of typical graphs in an hereditary family

A *graph property* P is called *hereditary* if every induced subgraph of a graph G with property P has property P, too. With no danger of confusion, we use the same notation P to denote a (hereditary) graph property and the family of all graphs that satisfy this property. Clearly, the properties that a graph G is a string graph ($G \in$ String) or that G is an intersection graph of plane convex sets ($G \in$ Conv) are hereditary. The same is true for the properties that G contains no subgraph, resp., no induced subgraph isomorphic to a fixed graph H.

It is a classic topic in extremal graph theory to investigate the typical structure of graphs in a specific hereditary family. This involves proving that almost all graphs in the family have a certain structural decomposition. This research is inextricably linked to the study of the growth rate of the function $|\mathrm{P}_n|$, also known as the *speed* of P, in two ways. Firstly, structural decompositions may give us bounds on the growth rate. Secondly, lower bounds on the growth rate help us to prove that the size of the exceptional family of graphs which fail to have a specific structural decomposition is negligible. In particular, we will both use a preliminary bound on the speed in proving our structural result about string graphs, and apply our theorem to improve the best known current bounds on the speed of the string graphs.

In a pioneering paper, Erdős, Kleitman, and Rothschild [ErKR76] approximately determined for every t the speed of the property that the graph contains no clique of size t. Erdős, Frankl, and Rödl [ErFR86] generalized this result as follows. Let H be a fixed graph with chromatic number $\chi(H)$. Then every graph of n vertices that does not contain H as a (not necessarily induced) subgraph can be made $(\chi(H)-1)$-partite by the deletion of $o(n^2)$ edges. This implies that the speed of the property that the graph contains no subgraph isomorphic to H is

$$2^{\left(1-\frac{1}{\chi(H)-1}+o(1)\right)\binom{n}{2}}. \tag{1}$$

Prömel and Steger [PrS92a, PrS92b, PrS93] established an analogous theorem for graphs containing no *induced subgraph* isomorphic to H. Throughout this paper, these graphs will be called *H-free*. To state their result, Prömel and Steger introduced the following key notion.

▶ **Definition 6.** A graph G is (r,s)-*colorable* for some $0 \le s \le r$ if there is a r-coloring of the vertex set $V(G)$, in which the first s color classes are cliques and the remaining $r-s$ color classes are independent sets. The *coloring number* $\chi_c(\mathrm{P})$ of a hereditary graph property P is the largest integer r for which there is an s such that all (r,s)-colorable graphs have property P. Consequently, for any $0 \le s \le \chi_c(\mathrm{P})+1$, there exists a $(\chi_c(\mathrm{P})+1,s)$-colorable graph that does not have property P.

The work of Prömel and Steger was completed by Alekseev [Al93] and by Bollobás and Thomason [BoT95, BoT97], who proved that the speed of any hereditary graph property P satisfies

$$|\mathrm{P}_n| = 2^{\left(1-\frac{1}{\chi_c(\mathrm{P})}+o(1)\right)\binom{n}{2}}. \tag{2}$$

The lower bound follows from the observation that for $\chi_c(\mathrm{P})=r$, there exists $s \le r$ such that all (r,s)-colorable graphs have property P. In particular, P_n contains all graphs whose vertex sets can be partitioned into s cliques and $r-s$ independent sets, and the number of such graphs is equal to the right-hand side of (2).

As for string graphs, Pach and Tóth [PaT06] proved that

$$\chi_c(\textsc{String}) = 4. \tag{3}$$

Hence, (2) immediately implies

$$|\textsc{String}_n| = 2^{\left(\frac{3}{4}+o(1)\right)\binom{n}{2}}. \tag{4}$$

If we want to tighten the above estimates, another idea of Prömel and Steger [PrS91] is instructive. They noticed that the vertex set of almost every C_4-free graph can be partitioned into a clique and an independent set, and no matter how we choose the edges between these two parts, we always obtain a C_4-free graph. Therefore, the speed of C_4-freeness is at most $(1+o(1))2^n 2^{\frac{1}{2}\binom{n}{2}}$, which is much better than the general bound $2^{\left(\frac{1}{2}+o(1)\right)\binom{n}{2}}$ that follows from (2). Almost all C_5-free graphs permit similar *"certifying partitions"*. It is an interesting open problem to decide which hereditary families permit such partitions and what can be said about the inner structure of the subgraphs induced by the parts. This line of research was continued by Balogh, Bollobás, and Simonovits [BaBS04, BaBS09, BaBS11]. The strongest result in this direction was proved by Alon, Balogh, Bollobás, and Morris [AlBBM11], who proved that for almost every graph with a hereditary property P, one can delete a small

fraction of the vertices in such a way that the rest can be partitioned into $\chi_c(\mathrm{P})$ parts with a very simple inner structure. This allowed them to replace the bound (2) by a better one:

$$|\mathrm{P}_n| = 2^{\left(1-\frac{1}{\chi_c(\mathrm{P})}\right)\binom{n}{2}+O(n^{2-\epsilon})}.$$

This will be the starting point of our analysis of string graphs. As we shall see, in the case of string graphs, our results allow us to replace the $2^{O(n^{2-\epsilon})}$ in this bound by $2^{\frac{9n}{4}+o(n)}$. See [BB11, KKOT15, RY17, ReSc17], for related results.

We need some notation. Following Alon *et al.*, for any integer $k > 0$, define $U(k)$ as a bipartite graph with vertex classes $\{1, \ldots, k\}$ and $\{I : I \subset \{1, ..., k\}\}$, where a vertex i in the first class is connected to a vertex I in the second if and only if $i \in I$. We think of $U(k)$ as a "universal" bipartite graph on $k + 2^k$ vertices, because for every subset of the first class there is a vertex in the second class whose neighborhood is precisely this subset.

As usual, the *neighborhood* of a vertex v of a graph G is denoted by $N_G(v)$ or, if there is no danger of confusion, simply by $N(v)$. For any disjoint subsets $A, B \subset V(G)$, let $G[A]$ and $G[A, B]$ denote the subgraph of G induced by A and the *bipartite* subgraph of G consisting of all edges of G running between A and B, respectively. The *symmetric difference* of two sets, X and Y, is denoted by $X \triangle Y$.

▶ **Definition 7.** Let k be a positive integer. A graph G is said to *contain* $U(k)$ if there are two disjoint subsets $A, B \subset V(G)$ such that the bipartite subgraph $G[A, B] \subseteq G$ induced by them is isomorphic to $U(k)$. Otherwise, with a slight abuse of terminology, we say that G is $U(k)$-free.

By slightly modifying the proof of the main result (Theorem 1) in [AlBBM11] and adapting it to string graphs, we obtain

▶ **Theorem 8.** *For any sufficiently large positive integer k and for any $\delta > 0$ which is sufficiently small in terms of k, there exist $\epsilon > 0$ and a positive integer b with the following properties.*

The vertex set V_n ($|V_n| = n$) of almost every string graph G can be partitioned into eight sets, $S_1, ...S_4, A_1, ..., A_4$, and a set B of at most b vertices such that

(a) *$G[S_i]$ is $U(k)$-free for every i ($1 \le i \le 4$);*

(b) *$|A_1 \cup A_2 ... \cup A_4| \le n^{1-\epsilon}$; and*

(c) *for every i ($1 \le i \le 4$) and $v \in S_i \cup A_i$ there is $a \in B$ such that*

$$|(N(v) \triangle N(a)) \cap (S_i \cup A_i)| \le \delta n.$$

In other words, for the right choice of parameters, almost all string graphs have a partition into 4 parts satisfying the following conditions. There is a set of sub-linear size in the number of vertices such that deleting its elements, the subgraphs induced by the parts are U(k)-free. Moreover, there is another set B of at most constantly many vertices such that the neighborhood of every vertex with respect to the part it belongs to is similar to the neighbourhood of some vertex in B. In the full version of the paper [PRY18], we sketch the proof of this result, indicating the places where we slightly deviate from the original argument in [AlBBM11].

3 String graphs vs. intersection graphs of convex sets – proof of Theorem 4

Instead of proving Theorem 4, we establish a somewhat more general result.

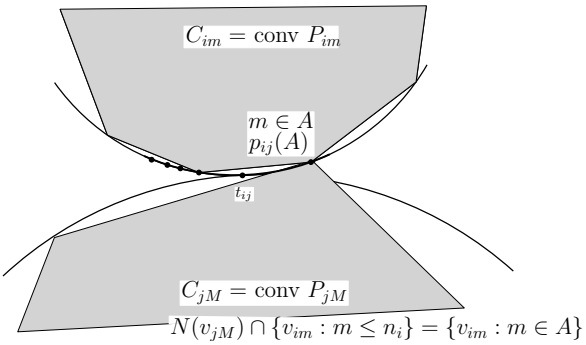


Figure 2 The point $p_{ij}(A)$ is included in P_{jM}.

▶ **Theorem 9.** *Given a planar graph H with labeled vertices $\{1, \dots, k\}$ and positive integers $n_1, \dots, n_k$, let $\mathrm{H}(n_1, \dots, n_k)$ denote the class of all graphs with $n_1 + \dots + n_k$ vertices that can be obtained from H by replacing every vertex $i \in V(H)$ with a clique of size n_i, and adding any number of further edges between pairs of cliques that correspond to pairs of vertices $i \neq j$ with $ij \in E(G)$.*

Then every element of $\mathrm{H}(n_1, \dots, n_k)$ is the intersection graph of a family of plane convex sets.

Proof. Fix any graph $G \in \mathrm{H}(n_1, \dots, n_k)$. The vertices of H can be represented by closed disks $D_1, \dots, D_k$ with disjoint interiors such that D_i and D_j are tangent to each other for some $i < j$ if and only if $ij \in E(H)$ (Koebe, [Ko36]). In this case, let $t_{ij} = t_{ji}$ denote the point at which D_i and D_j touch each other. For any i $(1 \le i \le k)$, let o_i be the center of D_i. Assume without loss of generality that the radius of every disk D_i is at least 1.

G has $n_1 + \dots + n_k$ vertices denoted by v_{im}, where $1 \le i \le k$ and $1 \le m \le n_i$. In what follows, we assign to each vertex $v_{im} \in V(G)$ a finite set of points P_{im}, and define C_{im} to be the convex hull of P_{im}. For every $i, 1 \le i \le k$, we include o_i in all sets P_{im} with $1 \le m \le n_i$, to make sure that for each i, all sets $C_{im}, 1 \le m \le n_i$ have a point in common, therefore, the vertices that correspond to these sets induce a clique.

Let $\varepsilon < 1$ be the *minimum* of all angles $\measuredangle t_{ij} o_i t_{il} > 0$ at which the arc between two consecutive touching points t_{ij} and t_{il} on the boundary of the same disc D_i can be seen from its center, over all $i, 1 \le i \le k$ and over all j and l. Fix a small $\delta > 0$ satisfying $\delta < \varepsilon^2/100$.

For every $i < j$ with $ij \in E(H)$, let γ_{ij} be a circular arc of length δ on the boundary of D_i, centered at the point $t_{ij} \in D_i \cap D_j$. We select 2^{n_i} distinct points $p_{ij}(A) \in \gamma_{ij}$, each representing a different subset $A \subseteq \{1, \dots, n_i\}$. A point $p_{ij}(A)$ will belong to the set P_{im} if and only if $m \in A$. (Warning: Note that the roles of i and j are not interchangeable!)

If for some $i < j$ with $ij \in E(H)$, the intersection of the neighborhood of a vertex $v_{jM} \in V(G)$ for any $1 \le M \le n_j$ with the set $\{v_{im} : 1 \le m \le n_i\}$ is equal to $\{v_{im} : m \in A\}$, then we include the point $p_{ij}(A)$ in the set P_{jM} assigned to v_{jM}, see Figure 2 for a sketch. Hence, for every $m \le n_i$ and $M \le n_j$, we have

$$v_{im}v_{jM} \in E(G) \iff P_{im} \cap P_{jM} \neq \emptyset.$$

In other words, the intersection graph of the sets assigned to the vertices of G is isomorphic to G.

It remains to verify that

$$v_{im}v_{jM} \in E(G) \iff C_{im} \cap C_{jM} \neq \emptyset.$$

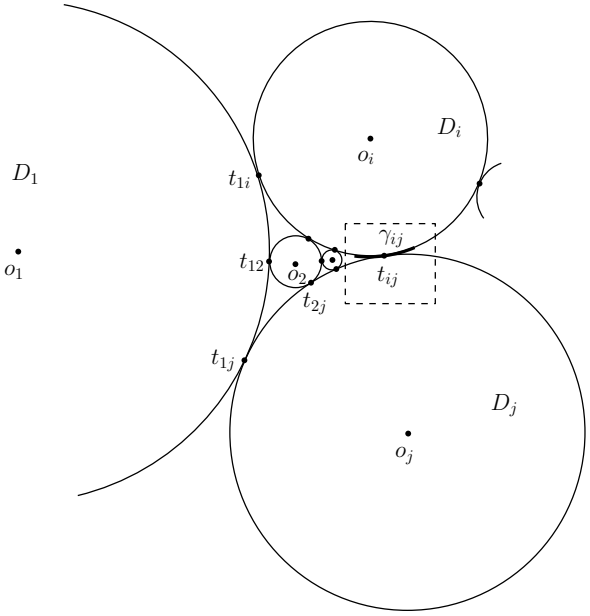


Figure 3 Tangent disks D_i and D_j touching at t_{ij}.

Suppose that the intersection graph of the set of convex polygonal regions

$$\{C_{im} : 1 \le i \le k \text{ and } 1 \le m \le n_i\}$$

differs from the intersection graph of

$$\{P_{im} : 1 \le i \le k \text{ and } 1 \le m \le n_i\}.$$

Assume first, for contradiction, that there exist i, m, j, M with $i < j$ such that D_i and D_j are tangent to each other and C_{jM} contains a point $p_{ij}(B)$ for which

$$B \neq N_{jM} \cap \{v_{im} : 1 \le m \le n_i\}. \tag{5}$$

Consider the unique point $p = p_{ij}(A) \in \gamma_{ij}$ that belongs to P_{jM}, that is, we have

$$A = N_{jM} \cap \{v_{im} : 1 \le m \le n_i\}.$$

Draw a tangent line ℓ to the arc γ_{ij} at point p. See Figure 3. The polygon C_{jM} has two sides meeting at p; denote the infinite rays emanating from p and containing these sides by r_1 and r_2. These rays either pass through o_j or intersect the boundary of D_j in a small neighborhood of the point of tangency of D_j with some other disk $D_{j'}$. Since δ was chosen to be much smaller than ε, we conclude that r_1 and r_2 lie entirely on the same side of ℓ where o_j, the center of D_j, is. On the other hand, all other points of γ_{ij}, including the point $p_{ij}(B)$ satisfying (5) lie on the opposite side of ℓ, which is a contradiction.

Essentially the same argument and a little trigonometric computation show that for every j and M, the set $C_{jM} \setminus D_j$ is covered by the union of some small neighborhoods (of radius $< \varepsilon/10$) of the touching points t_{ij} between D_j and the other disks D_i. This, together with the assumption that the radius of every disk D_i is at least 1 (and, hence, is much larger than ε and δ) implies that C_{jM} cannot intersect any polygon C_{im} with $i \neq j$, for which D_i and D_j are not tangent to each other. ◀

Applying Theorem 9 to the graph obtained from K_5 by deleting one of its edges, Theorem 4 follows.

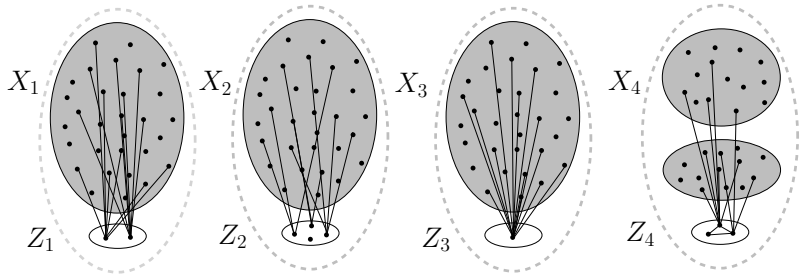


Figure 4 A sketch of a typical string graph as in Theorem 10. The edges between the parts are not drawn. The sets shaded grey are cliques.

4 Strengthening Theorem 8

In this section, we strengthen Theorem 8 in two different ways. To avoid confusion, in the formulation of our new theorem, we use X_i in place of S_i and Z_i in place of A_i. We will see that we can insist that the four parts of the partition have approximately the same size. Secondly, we can guarantee that X_1, X_2, and X_3 are cliques and X_4 induces the disjoint union of two cliques. More precisely, setting $Z = Z_1 \cup Z_2 ... \cup Z_4$, we prove the following result, which is similar in flavour to a result in [ReSc17].

▶ **Theorem 10.** *For every sufficiently small δ, there are $\gamma > 0, b > 4 + \frac{2}{\delta}$ with the following property. For almost every string graph G on V_n, there is a partition of V_n into $X_1, ..., X_4, Z_1, ..., Z_4$ such that for some set B of at most b vertices the following conditions are satisfied:*

- **(I)** *$G[X_1]$, $G[X_2]$, and $G[X_3]$ are cliques and $G[X_4]$ induces the disjoint union of two cliques.*
- **(II)** *$|Z_1 \cup Z_2 \cup Z_3 \cup Z_4| \leq n^{1-\gamma}$,*
- **(III)** *for every i $(1 \leq i \leq 4)$ and every $v \in X_i \cup Z_i$, there exists $a \in B$ such that*

$$|(N(v) \triangle N(a)) \cap (X_i \cup Z_i)| \leq \delta n,$$

- **(IV)** *for every i $(1 \leq i \leq 4)$, we have $\left||Z_i \cup X_i| - \frac{n}{4}\right| \leq n^{1-\gamma}$.*

See Figure 4 for an illustration of Theorem 10.

For the proof of Theorem 10 we need the following statement which is a slight generalization of Lemma 3.2 in [PaT06], and it can be established in precisely the same way, details are given in the full version of the paper [PRY18].

▶ **Lemma 11.** *Let H be a graph on the vertex set $\{v_1, \ldots, v_5\} \cup \{v_{ij} : 1 \leq i \neq j \leq 5\}$, where $v_{ij} = v_{ji}$ and every v_{ij} is connected by an edge to v_i and v_j. The graph H may have some further edges connecting pairs of vertices (v_{ij}, v_{ik}) with $j \neq k$. Then H is not a string graph.*

▶ **Corollary 12.** *For each of the following types of partition, there exists a non-string graph whose vertex set can be partitioned in the specified way:*

- **(a)** *2 stable (that is, independent) sets each of size at most 10;*
- **(b)** *4 cliques each of size at most five and a vertex;*
- **(c)** *3 cliques each of size at most five and a stable set of size 3;*
- **(d)** *3 cliques each of size at most five and a path with three vertices;*
- **(e)** *2 cliques both of size at most five and 2 graphs that can be obtained as the disjoint union of a point and a clique of size at most 3.*

See Figure 5 for an illustration of Corollary 12.

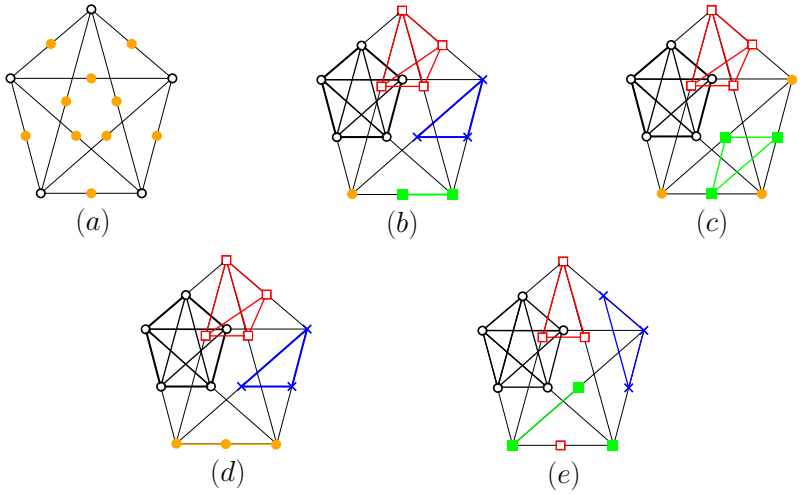


Figure 5 Possible partitions of a non-string graph.

Proof of Theorem 10. We choose k sufficiently large and then $\delta < \frac{1}{40}$ sufficiently small in terms of k. We choose $\epsilon, b > 0$ such that Theorem 8 holds for this choice of k and δ and so that ϵ is less than the ρ of Lemma 14 for this choice of k. We set $\gamma = \frac{\epsilon}{10}$ and consider n large enough to satisfy certain implicit inequalities below. We know that the subset $\mathcal{S}(k,\delta)_n$ of STRING_n, consisting of those graphs for which there is a set B of at most b vertices and a partition into S_i and A_i satisfying (a),(b), and (c) set out in Theorem 8, contains almost every string graph. We call such a partition, *certifying*. We need to show that almost every graph in $\mathcal{S}(k,\delta)_n$ has a certifying partition for which we can repartition $S_i \cup A_i$ into $X_i \cup Z_i$ so that (I),(II), and (IV) all hold (that (III) holds, is simply Theorem 8 (c) and $S_i \cup A_i = X_i \cup Z_i$).

We prove this fact via a sequence of lemmas. In doing so, for a specific partition, we let $m = m(A_1 \cup S_1, A_2 \cup S_2, A_3 \cup S_3, A_4 \cup S_4)$ be the number of pairs of vertices not lying together in some $A_i \cup S_i$. The first lemma gives us a lower bound on $|\mathcal{S}(k,\delta)_n|$, obtained by simply counting the number of graphs which permits a partition into four cliques all of size within one of $\frac{n}{4}$. Its proof is given in the full version of the paper [PRY18].

▶ **Lemma 13.** $|\mathcal{S}(k,\delta)_n| \geq 2^{\frac{3\binom{n}{2}}{4}}$.

The second gives us an upper bound on the number of choices for $G[S_i]$ for graphs G in $\mathcal{S}(k,\delta)_n$ for which $S_1, S_2, S_3, S_4, A_1, A_2, A_3, A_4$ is a certifying partition. It is Corollary 8 in [AlBBM11].

▶ **Lemma 14.** *For every k, there is a positive ρ such that for every sufficiently large l, the number of $U(k)$-free graphs with l vertices is less than $2^{l^{2-\rho}}$.*

Next we prove:

▶ **Lemma 15.** *The number of graphs in $\mathcal{S}(k,\delta)_n$ which have a certifying partition such that for some i, $||A_i \cup S_i| - \frac{n}{4}| > n^{1-\gamma}$ is $o(|\mathcal{S}(k,\delta)_n|)$.*

Proof. The number of choices for a partition of V_n into $S_1, S_2, S_3, S_4, A_1, A_2, A_3, A_4$ is at most 8^n. If this partition demonstrates that S_i is $U(k)$-free and n is large, Lemma 14 tells us that there are only $2^{n^{2-\epsilon}}$ choices for $G[S_i]$. The number of choices for the edges out of each vertex of A_i is 2^{n-1}. So, since $|A_i|$ is at most $n^{1-\epsilon}$, we know there are at most $2^{n^{2-\epsilon}}$ choices for the edges out of A_i. It follows that there are at most $2^{11(n^{2-\epsilon})}$ choices for

our partition and the graphs $G[S_1 \cup A_1], ..., G[S_4 \cup A_4]$ over all G in $\mathcal{S}(k,\delta)_n$ which can be certified using this partition. Furthermore,the number of graphs in $\mathcal{S}(k,\delta)_n$ permitting such a certifying choice is at most 2^m. Since, $|\mathcal{S}(k,\delta)_n| \geq 2^{\frac{3\binom{n}{2}}{4}}$, it follows that almost every graph G in $\mathcal{S}(k,\delta)_n$ has no certifying partition for which $m < \frac{3\binom{n}{2}}{4} - 12(n^{2-\epsilon})$. The desired result follows. ◀

Setting $l = l_n = \lceil n^{1-\frac{\epsilon}{7}} \rceil$, we have the following.

▶ **Lemma 16.** *The number of graphs in $\mathcal{S}(k,\delta)_n$ which have a certifying partition for which there are distinct i and j such that both S_i and S_j contain l disjoint independent sets of size 10 is $o(|\mathcal{S}(k,\delta)_n|)$.*

Proof. Consider a choice of certifying partition and induced subgraphs H_1, H_2, H_3, H_4 where $V(H_i) = A_i \cup S_i$. By Corollary 12(a), for any pair of independent sets of size 10, at least one of the 2^{100} choices of edges between the sets yields a bipartite non-string graph. Thus, the number of choices for edges between the partitions which extend our choice to yield a graph in $String_n$ is at most $2^m(1 - \frac{1}{2^{100}})^{l^2}$. Since $m < \frac{3\binom{n}{2}}{4}$ and $l^2 = \omega(n^{2-\frac{\epsilon}{2}})$, it follows that for almost every graph in $\mathcal{S}(k,\delta)_n$, almost every certifying partition does not contain two distinct such i and j. ◀

Ramsey theory tells us that if a graph J does not contain l disjoint stable sets of size 10, it contains $|V(J)| - 10(l-1) - 2^{15}$ disjoint cliques of size 5. Combining applications of this fact to three of the $G[S_i]$, Corollary 11(c), and an argument similar to that used in the proof of Lemma 16 allows us to prove the following lemma. Details can be found in the full version of the paper [PRY18].

▶ **Lemma 17.** *The number of graphs G in $\mathcal{S}(k,\delta)_n$ which have a certifying partition for which there is an $i = i(G)$ such that S_i does not contain l disjoint cliques of size 5 is $o(|\mathcal{S}(k,\delta)_n|)$*

With this lemma in hand, we can mimic the argument used in its proof to obtain the following two lemmas. In doing so, we apply Corollary 11 (c),(d), and (e).

▶ **Lemma 18.** *The number of graphs G in $\mathcal{S}(k,\delta)_n$ which have a certifying partition for which there is an $i = i(G)$ such that S_i contains l disjoint sets of size three each inducing a stable set or a path is $o(|\mathcal{S}(k,\delta)_n|)$.*

▶ **Lemma 19.** *The number of graphs G in $\mathcal{S}(k,\delta)_n$ which have a certifying partition for which there are two distinct i such that S_i contains l disjoint sets of size four each inducing the disjoint union of a vertex and a triangle is $o(|\mathcal{S}(k,\delta)_n|)$.*

Combining these lemmas, and possibly permuting indices, we see that almost every graph in $\mathcal{S}(k,\delta)_n$ has a certifying partition for which for every $i \leq 4$ we have $||Z_i \cup X_i| - \frac{n}{4}| \leq n^{1-\gamma}$, no S_i contains more than l sets inducing a path of length three or a stable set of size three, and for every $k \leq 3$, S_k does not contain l disjoint sets inducing the disjoint union of a vertex and a triangle. For each such graph, we consider such a partition. For all $i < 4$, we let Z_i be the union of A_i and a maximum family of disjoint sets in X_i each inducing a path of length 3, a stable set of size three, or the disjoint union of a triangle and a vertex. We let Z_4 be the union of A_4 and a maximum family of disjoint sets in X_4 each inducing a path of length three or a stable set of size three. We set $X_i = S_i - Z_i$. ◀

5 Completing the proof of Theorem 1

In this section, we prove our main result. By a *great* partition of G we mean a partition of its vertex set into X_1, X_2, X_3, X_4 such that for $i \leq 3$, X_i is a clique and X_4 is the disjoint union of two cliques. We call a graph *great* if it has a great partition and *mediocre* otherwise. Theorem 1 simply states that almost every string graph G on V_n is great.

Thus, we are trying to show that almost every string graph has a partition into sets $X_1, X_2, X_3, X_4, Z_1, Z_2, Z_3, Z_4$ satisfying Theorem 10 (I) with the sets Z_i empty. We choose δ so small that Theorem 10 holds and δ also satisfies certain inequalities implicitly given below. We apply Theorem 10 and obtain that for some positive γ and b, for almost every graph in $\textsc{String}_n$ there is a partition of V_n into $X_1, ..., X_4, Z_1, .., Z_4$ satisfying (I), (II), (III), and (IV). Note that if we reduce γ the theorem remains true. We insist that γ is at most $\frac{1}{64000000}$. We call such partitions *good*. We need to show that the number of mediocre string graphs on V_n with a good partition is of smaller order than the number of great graphs on V_n.

The following result tells us that the number of great graphs on V_n is of the same order as the number of great partitions of graphs on V_n.

▶ **Claim 20.** *The ratio between the number of great partitions of graphs on V_n and the number of graphs which permit such partitions is $6 + o(1)$.*

So, it is sufficient to show that the number of mediocre string graphs with a good partition on V_n is of smaller order than the number of graphs with a great partition on V_n. In doing so, we consider each partition separately. For every partition $\mathcal{Y} = (Y_1, Y_2, Y_3, Y_4)$ of V_n we say that a good partition satisfying (I)-(IV) with $Y_i = X_i \cup Z_i$ for every i is $\mathcal{Y}$-good. We prove:

▶ **Claim 21.** *For every partition $\mathcal{Y} = (Y_1, Y_2, Y_3, Y_4)$ of V_n, the number of graphs which permit a great partition with $X_i = Y_i$ for every i is of larger order then the size of the set $\mathcal{F} = \mathcal{F}_{\mathcal{Y}}$ of mediocre string graphs which permit a $\mathcal{Y}$-good partition.*

To complete the proof of Theorem 1 we need to show that our two claims hold.

Before doing so, we deviate momentarily and discuss the speed of the string graphs. Combining Theorem 1 and Claim 20, we see that the ratio of the size of $|\textsc{String}_n|$ over the number of ordered great partitions of graphs on V_n is $\frac{1}{6} + o(1)$, so we need only count the latter. There are 2^{2n} ordered partitions of V_n into $Y_1, ..., Y_4$, and there are $2^{m+|Y_4|}$ graphs for which this is a great partition, where, as before, $m = m(Y_1, Y_2, Y_3, Y_4)$ is the number of pairs of vertices not lying together in some Y_i. This latter term is at most $2^{\frac{3\binom{n}{2}}{4}+\frac{n}{4}}$, which gives us the claimed upper bound on the speed of string graphs. Furthermore, a simple calculation of the 2^{2n} ordered 4-partitions of V_n shows that there is an $\Omega(\frac{1}{n^{\frac{3}{2}}})$ proportion where no two parts differ in size by more than one. This gives us the claimed lower bound.

We now prove our two claims. In proving both, we exploit the fact that if a string graph has a great partition and we fix the subgraph induced by the parts of the partition, then any choice we make for the edges between the sets X_i will yield another string graph permitting the same great partition.

This fact implies that the edge arrangements between the partition elements of a graph permitting a particular great partition are chosen uniformly at random and, hence, are unlikely to lead to a graph permitting some other great partition. This allows us to prove Claim 20, which we do in the full version of the paper [PRY18].

Proof of Claim 21. Let m be the number of pairs of vertices not contained in a partition element and note that there are exactly $(2^{|Y_4|-1})$ choices for $G[Y_4]$ for a graph for which $\mathcal{Y}$ is a great partition, and hence $2^m(2^{|Y_4|-1})$ graphs for which $\mathcal{Y}$ is a great partition.

Our approach is to show that while there may be more choices for the $G[Y_i]$ for mediocre graphs for which $\mathcal{Y}$ is a good partition, for each such choice we have many fewer than 2^m choices for mediocre string graphs extending these subgraphs.

We note that by the definition of good, we need only consider partitions such that each Y_i has size $\frac{n}{4} + o(n)$.

Let $G \in \mathcal{F}$ and let $P(G)$ be the projection of G on the sets (Y_1, Y_2, Y_3, Y_4), that is, the disjoint union of the sets $G[Y_1], G[Y_2], G[Y_3]$, and $G[Y_4]$.

Now, (I) of Theorem 10 bounds the number of choices for $G[Y_i]$ by 1 if $i < 3$ and $2^{|Y_4|}$ if $i = 4$. Furthermore, (III) bounds the number of edges out of Z_i in terms of its size and (II) bounds its size. Putting this all together we obtain the following lemma. Its proof can be found in the full version of the paper [PRY18].

▶ **Lemma 22.** *Let (Y_1, Y_2, Y_3, Y_4) be a partition of V_n, the number of possible projections on (Y_1, Y_2, Y_3, Y_4) of graphs in $\mathcal{F}$ is $o(2^{nb+1+\sqrt{\delta}n|Z|}) = o(2^{|Y_4|-1} \cdot 2^{\sqrt{\delta}n^{2-\gamma}})$.*

For a mediocre graph G in $\mathcal{F}$, we call a set D *versatile* if for each $i \in [4]$ with $Y_i \cap D = \emptyset$, there is clique C_i in Y_i such that for all subsets D' of D there are $\frac{n}{\log n}$ vertices of C_i which are adjacent to all elements of D' and to none of $D \setminus D'$.

▶ **Lemma 23.** *The number of mediocre string graphs in $\mathcal{F}$ such that for some i there is a versatile subset T_i of 3 vertices of Y_i inducing a path or a stable set of size three,is $o(2^m)$.*

Proof. To begin, we count the number of mediocre graphs which extend a given projection on (Y_1, Y_2, Y_3, Y_4) where T_i induces such a graph. We first expose the edges from Y_i to determine if T_i is versatile and then count the number of choices for the remaining edges between the partition elements. If T_i is versatile we choose cliques C_k which show this is the case.

By Corollary 12 (c) or (d), there is a non-string graph J whose vertex set can be partitioned into 3 cliques of size at most five, and a graph J_i isomorphic to the subgraph of the projection induced by T_i. We label these three cliques as J_k for $k \in \{1, 2, 3, 4\} - \{i\}$ and let f be an isomorphism from J_i to T_i. For each vertex $v \in V(J_k)$, let $N(v) = f(N_J(v) \cap V(J_i))$ and Z_v be those vertices of C_k whose neighbourhhod on T_i is $N(v)$. Now, since $|Z_v| \geq \frac{n}{\log n}$ for all v in each $V(J_k)$, for each $k \neq i$, we can choose $n' = \lceil \frac{n}{10 \log n} \rceil$ cliques of size at most five $C_1^k, ..., C_{n'}^k$ such that there is bijection $h_{k,l}$ from J_k to C_l^k with $h_{k,l}(v) \in Z_v$ for every $v \in J_k$.

If we choose our cliques in this way then for any set of three cliques $\{C_{i(k)}^k | k \neq i\}$ there is a choice of edges between the cliques which would make the union of these three cliques with T_i induce J. Thus, there is one choice of edges between the cliques which cannot be used in any extension of H to a string graph. Mimicking an earlier argument, this implies that the number of choices for edges between the partition elements which extend H to a string graph is at most $2^{m - \frac{n^2}{\log^3 n}}$. By the bound in Lemma 22 on the number of possible projections, the desired result follows. ◀

Using Corollary 12 (e) in places of (c) & (d), we can (and do in the in the full version of the paper [PRY18]) prove an analogous result for sets of size 8 intersecting two partition elements. To state it we need a definition. A graph J is *extendible* if there is some non-string graph whose vertex set can be partitioned into two cliques of size five and a set inducing J.

▶ **Lemma 24.** *The number of mediocre string graphs in $\mathcal{F}$ such that for some distinct i and k there are subsets T_i of Y_i and T_k of Y_k, both of size four, whose union is both versatile and induces an extendible graph is $o(2^m)$.*

For every mediocre string graph G in $\mathcal{F}$, we choose a maximum family $\mathcal{W} = \mathcal{W}_G$ of disjoint sets each of which is either (a) contained in some Y_i and induces one of a stable set of size three or a path of length three, or (b) contains exactly four vertices from each of two distinct partition elements and is extendible. For every such choice we count the number of elements of $\mathcal{F}$ whose projection yields the given choice of $\mathcal{W}$.

Now, by the definition of a good partition, each Y_k contains a clique C_k containing half the vertices of X_k and hence at least $\frac{n}{10}$ vertices. Lemmas 23 and 24 imply that we can restrict our attention to graphs for which for any subset T in $\mathcal{W}$, there is a subset N of T and a j with Y_j disjoint from T such that there are fewer than $\frac{n}{logn}$ vertices of C_k which are adjacent to all of N and none of $T - N$. This implies that the number of choices for the edges from T to other partition elements is $o(2^{\frac{3n|T|}{4} - \frac{n}{10000}})$.

Every element of $\mathcal{W}$ must intersect Z, so that $|\mathcal{W}| \leq |Z|$. Set $W^* = \cup_{W \in \mathcal{W}} W$, and let $Y_i' = Y_i - W^*$. Note that for every i, Y_i' has more than $\frac{n}{5}$ vertices and $G[Y_i']$ is the disjoint union of two cliques. Given a choice of $\mathcal{W}$, the number of choices for projections on $V_n \setminus W^*$ is less than 2^n. Mimicking the proof of Lemma 22, the number of choices for the vertices of W^*, and the edges of G from the vertices in W^* which remain within the partition elements of $\mathcal{Y}$ is $O(2^{bn + \sqrt{\delta}|W^*|n})$. Combining this with the result of the last paragraph yields:

▶ **Lemma 25.** *There is a constant C such that the number of mediocre string graphs in $\mathcal{F}$ for which $|\mathcal{W}| > C$ is $o(2^{m+|Y_4|})$.*

So, we can restrict our attention to mediocre graphs which have a partition for which $|\mathcal{W}| \leq C$. Similar tradeoffs allow us to handle them. Full details are found in the full version of the paper [PRY18]. ◀

References

Al93 V. E. Alekseev. On the entropy values of hereditary classes of graphs, *Discrete Math. Appl.* **3** (1993), 191–199.

AlBBM11 N. Alon, J. Balogh, B. Bollobás, and R. Morris. The structure of almost all graphs in a hereditary property, *J. Combin. Theory Ser. B* **101(2)** (2011), 85–110.

BaBS04 J. Balogh, B. Bollobás, and M. Simonovits. On the number of graphs without forbidden subgraph, *J. Combin. Theory Ser. B* **91** (2004), 1–24.

BaBS09 J. Balogh, B. Bollobás, and M. Simonovits. The typical structure of graphs without given excluded subgraphs, *Random Structures Algorithms* **34** (2009), 305–318.

BaBS11 J. Balogh, B. Bollobás, and M. Simonovits. The fine structure of octahedron-free graphs, *J. Combin. Theory Ser. B* **101(2)** (2011), 67–84.

BB11 J. Balogh and J. Butterfield. Excluding induced subgraphs: critical graphs, *Random Structures and Algorithms* **38** (2011), 100–120.

Be59 S. Benzer. On the topology of the genetic fine structure, *Proc. Nat. Acad. Sci.* **45** (1959), 1607–1620.

BoT95 B. Bollobás and A. Thomason. Projections of bodies and hereditary properties of hypergraphs, *Bull. Lond. Math. Soc.* **27** (1995), 417–424.

BoT97 B. Bollobás and A. Thomason. Hereditary and monotone properties of graphs, *The Mathematics of Paul Erdős, Vol. II, R. L. Graham and J. Nešetřil (Eds.)* **14** (1997), 70–78.

Ch34 Ch. Chojnacki (A. Hanani). Über wesentlich unplättbare Kurven im dreidimensionalen Raume, *Fund. Math.* **23** (1934), 135–142.

ErFR86 P. Erdős, P. Frankl, and V. Rödl. The asymptotic number of graphs not containing a fixed subgraph and a problem for hypergraphs having no exponent, *Graphs Combin.* **2** (1986), 113–121.

ErKR76 P. Erdős, D. J. Kleitman, B. L. Rothschild. Asymptotic enumeration of K_n-free graphs, *International Colloquium on Combinatorial Theory, Atti dei Convegni Lincei* **17** (1976), 19–27.

JaU17 S. Janson and A. J. Uzzell. On string graph limits and the structure of a typical string graph, *J. Graph Theory* **84** (2017), 386–407.

KKOT15 J. Kim, D. Kuhn, D. Osthus, T. Townsend. Forbidding induced even cycles in a graph: typical structure and counting, `http://arxiv.org/abs/1507.04944` (2015).

Ko36 P. Koebe. Kontaktprobleme der Konformen Abbildung, *Ber. Sachs. Akad. Wiss. Leipzig, Math.–Phys. Kl.*, **88** (1936), 141–164.

Kr91 J. Kratochvíl. String graphs II: recognizing string graphs is NP-hard, *J. Combin. Theory Ser. B* **52** (1991), 67–78.

KrMa91 J. Kratochvíl and J. Matoušek: String graphs requiring exponential representations, *J. Combin. Theory Ser. B* **53** (1991), 1–4.

PaT06 J. Pach and G. Tóth. How many ways can one draw a graph?, *Combinatorica* **26** (2006), 559–576.

PRY18 J. Pach, B. Reed and Y. Yuditsky. Almost all string graphs are intersection graphs of plane convex sets, `http://arxiv.org/abs/1803.06710`.

PrS91 H. J. Prömel and A. Steger. Excluding Induced Subgraphs I: Quadrilaterals, *Random Structures and Algorithms.* **2** (1991), 53–79.

PrS92a H. J. Prömel and A. Steger. Almost all Berge graphs are perfect, *Combin., Probab. & Comp.* **1** (1992), 53–79.

PrS92b H. J. Prömel and A. Steger. Excluding induced subgraphs. III. A general asymptotic, *Random Structures Algorithms 3(1)* (1992), 19–31.

PrS93 H. J. Prömel and A. Steger. Excluding induced subgraphs II: Extremal graphs, *Discrete Appl. Math.* **44** (1993) 283–294.

ReSc17 B. Reed and A. Scott. The typical structure of an H-free graph when H is a cycle, *manuscript.*

RY17 B. Reed and Y. Yuditsky. The typical structure of H-free graphs for H a tree, *manuscript.*

ScSeSt03 M. Schaefer, E. Sedgwick, and D. Štefankovič. Recognizing string graphs in NP. *Special issue on STOC 2002 (Montreal, QC), J. Comput. System Sci.* **67** (2003), 365–380.

ScSt04 M. Schaefer and D. Štefankovič. Decidability of string graphs, *J. Comput. System Sci.* **68** (2004), 319–334.

Si66 F. W. Sinden. Topology of thin film RC-circuits, *Bell System Technological Journal* (1966), 1639–1662.

Tu70 W. T. Tutte. Toward a theory of crossing numbers, *J. Combinatorial Theory* **8** (1970), 45–53.

An Improved Bound for the Size of the Set $A/A + A$

Oliver Roche-Newton[1]
Johann Radon Institute for Computational and Applied Mathematics (RICAM)
69 Altenberger Straße, Linz, Austria
o.rochenewton@gmail.com

Abstract

It is established that for any finite set of positive real numbers A, we have

$$|A/A + A| \gg \frac{|A|^{\frac{3}{2}+\frac{1}{26}}}{\log^{5/6}|A|}.$$

2012 ACM Subject Classification Mathematics of computing → Combinatoric problems

Keywords and phrases sum-product estimates, expanders, incidence theorems, discrete geometry

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.69

Acknowledgements I am grateful to Chun-Yen Shen and Ilya Shkredov for helpful conversations.

1 Introduction

Given a set A, we define its *sum set*, *product set* and *ratio set* as

$$A + A := \{a + b : a, b \in A\}, \quad AA := \{ab : a, b \in A\}, \quad A/A := \{a/b : a, b \in A, b \neq 0\}.$$

respectively. It was conjectured by Erdős and Szemerédi that, for any finite set A of integers, at least one of the sum set or product set has near-quadratic growth. Solymosi [9] used a beautiful and elementary geometric argument to prove that, for any finite set $A \subset \mathbb{R}$,

$$\max\{|A + A|, |AA|\} \gg \frac{|A|^{4/3}}{\log^{1/3}|A|}. \tag{1}$$

Recently, a breakthrough for this problem was achieved by Konyagin and Shkredov [3]. They adapted and refined the approach of Solymosi, whilst also utilising several other tools from additive combinatorics and discrete geometry in order to prove that

$$\max\{|A + A|, |AA|\} \gg |A|^{\frac{4}{3}+\frac{1}{20598}-o(1)}. \tag{2}$$

Further refinements in [4], [6] and very recently [7] have improved this exponent to $4/3 + 5/5277 + o(1)$. See [3], [4], [6], [7] and the references contained therein for more background on the sum-product problem.

In this paper the related problem of establishing lower bounds for the sets

$$AA + A := \{ab + c : a, b, c \in A\}, \qquad A/A + A := \{a/b + c : a, b, c \in A, b \neq 0\}$$

[1] Oliver Roche-Newton was supported by the Austrian Science Fund (FWF): Project F5511-N26, which is part of the Special Research Program "Quasi-Monte Carlo Methods: Theory and Applications" as well as by FWF Project P 30405-N32.

are considered. It is believed, in the spirit of the Erdős-Szemerédi conjecture, that these sets are always large. It was conjectured by Balog [1] that, for any finite set A of real numbers, $|AA + A| \geq |A|^2$. In the same paper, he proved the following result in that direction:

▶ **Theorem 1.** *Let A and B be finite sets of positive real numbers. Then*

$$|AB + A| \gg |A||B|^{1/2}.$$

In particular,

$$|AA + A| \gg |A|^{3/2} \quad |A/A + A| \gg |A|^{3/2}.$$

The proof of Theorem 1 uses similar elementary geometric arguments to those of [9]. In fact, one can obtain the same bound by a straightforward application of the Szemerédi-Trotter Theorem (see [10, Exercise 8.3.3]).

Some progress in this area was made by Shkredov [8], who built on the approach of Balog in order to prove the following result:

▶ **Theorem 2.** *For any finite set A of positive real numbers,*

$$|A/A + A| \gg \frac{|A|^{\frac{3}{2}+\frac{1}{82}}}{\log^{\frac{2}{41}} |A|}, \tag{3}$$

The main result of this paper is the following improvement on Theorem 2:

▶ **Theorem 3.** *Let A be a finite set of positive reals. Then*

$$|A/A + A| \gg \frac{|A|^{\frac{3}{2}+\frac{1}{26}}}{\log^{5/6} |A|}.$$

For the set $AA + A$ the situation is different, and it has proven rather difficult to beat the threshold exponent of 3/2. A detailed study of this set can be found be in a recent paper of the author, Ruzsa, Shen and Shkredov [5]. However, the corresponding problem for sets of integers is resolved, up to constant factors, thanks to a nice argument of George Shakan.[2]

1.1 Sketch of the proof of Theorem 3

The proof is a refined version of the argument used by Balog to prove Theorem 1. Balog's argument goes roughly as follows:

Consider the point set $A \times A$ in the plane. Cover this point set by lines through the origin. Let us assume for simplicity that all of these lines are equally rich, so we have k lines with $|A|^2/k$ points on each line. Label the lines $l_1, l_2, \ldots, l_k$ in increasing order of steepness. Note that if we take the vector sum of a point on l_i with a point on l_{i+1}, we obtain a point which has slope in between those of l_i and l_{i+1}. The aim is to show that many elements of $(A/A + A) \times (A/A + A)$ can be obtained by studying vector sums from neighbouring lines.

Indeed, for any $1 \leq i \leq k - 1$, consider the sum set

$$\{(b/a, c/a) + (d, e) : a \in A, (b, c) \in (A \times A) \cap l_i, (d, e) \in (A \times A) \cap l_{i+1}\}.$$

[2] See http://mathoverflow.net/questions/168844/sum-and-product-estimate-over-integers-rationals-and-reals, where this argument first appeared.

There are at least $|A|$ choices for $(b/a, c/a)$ and at least $|A|^2/k$ choices for (d, e). Since all of these sums are distinct, we obtain at least $|A|^3/k$ elements of $(A/A + A) \times (A/A + A)$ lying in between l_i and l_{i+1}. Summing over all $1 \leq i \leq k-1$, it follows that

$$|A/A + A|^2 \gg |A|^3.$$

There are two rather crude steps in this argument. The first is the observation that there are at least $|A|$ choices for the point $(b/a, c/a)$. In fact, the number of points of this form is equal to the cardinality of product set of A and a set of size $|A|^2/k$. This could be as small as $|A|$, but one would typically expect it to be considerably larger. This extra information was used by Shkredov [8] in his proof of (3).

The second wasteful step comes at the end of the argument, when we only consider sums coming from pairs of lines which are neighbours. This means that we consider only $k-1$ pairs of lines out of a total of $\binom{k}{2}$. A crucial ingredient in the proof of (2) was the ability to find a way to count sums coming from more than just neighbouring lines.

The proof of Theorem 3 deals with these two steps more efficiently. Ideas from [8] are used to improve upon the first step, and then ideas from [3] improve upon the second step. We also make use of the fact that the set A/A is invariant under the function $f(x) = 1/x$, which allows us to use results on convexity and sumsets of Elekes, Nathanson and Ruzsa [2] in order to get a better exponent in Theorem 3.

2 Notation and preliminary results

Throughout the paper, the standard notation $\ll, \gg$ as well as O, Ω is applied to positive quantities in the usual way. Saying $X \gg Y$ or $X = \Omega(Y)$ means that $X \geq cY$, for some absolute constant $c > 0$.

The main tool is the Szemerédi-Trotter Theorem.

▶ **Theorem 4.** *Let P be a finite set of points in $\mathbb{R}^2$ and let L be a finite set of lines. Then*

$$|\{(p, l) \in P \times L : p \in l\}| \ll (|P||L|)^{2/3} + |P| + |L|.$$

Define

$$d(A) = \min_{C \neq \emptyset} \frac{|AC|^2}{|A||C|}. \tag{4}$$

We will need the following consequence of the Szemerédi-Trotter Theorem, which is [3, Corollary 8].

▶ **Lemma 5.** *Let A_1, A_2 and A_3 be finite sets of real numbers and let α_1, α_2 and α_3 be arbitrary non-zero real numbers. Then the number of solutions to the equation*

$$\alpha_1 a_1 + \alpha_2 a_2 + \alpha_3 a_3 = 0,$$

such that $a_1 \in A_1$, $a_2 \in A_2$ and $a_3 \in A_3$, is at most

$$C \cdot d^{1/3}(A_1)|A_1|^{1/3}|A_2|^{2/3}|A_3|^{2/3},$$

for some absolute constant C.

Another application of (a variant of) the Szemerédi-Trotter Theorem is the following result of Elekes, Nathanson and Ruzsa [2]:

▶ **Theorem 6.** *Let $f : \mathbb{R} \to \mathbb{R}$ be a strictly convex or concave function and let $X, Y, Z \subset \mathbb{R}$ be finite. Then*

$$|f(X) + Y||X + Z| \gg |X|^{3/2}|Y|^{1/2}|Z|^{1/2}.$$

In particular, this theorem can be applied with $f(x) = 1/x$, $X = A/A$, $Y = Z = A$, using the fact that $f(A/A) = A/A$, to obtain the following corollary:

▶ **Corollary 7.** *For any finite set $A \subset \mathbb{R}$,*

$$|A/A + A| \gg |A/A|^{3/4}|A|^{1/2}.$$

3 Proof of main theorem

Recall that the aim is to prove the inequality

$$|A/A + A| \gg \frac{|A|^{\frac{3}{2}+\frac{1}{26}}}{\log^{1/2}|A|}.$$

Consider the point set $A \times A$ in the plane. At the outset, we perform a dyadic decomposition, and then apply the pigeonhole principle, in order to find a large subset of $A \times A$ consisting of points lying on lines through the origin which contain between τ and 2τ points, where τ is some real number.

Following the notation of [3], for a real number λ, define

$$\mathcal{A}_\lambda := \left\{(x, y) \in A \times A : \frac{y}{x} = \lambda\right\},$$

and its projection onto the horizontal axis,

$$A_\lambda := \{x : (x, y) \in \mathcal{A}_\lambda\}.$$

Note that $|A_\lambda| = |A \cap \lambda A|$ and

$$\sum_\lambda |A_\lambda| = |A|^2. \tag{5}$$

Let S_τ be defined by

$$S_\tau := \{\lambda : \tau \leq |A \cap \lambda A| < 2\tau\}.$$

After dyadically decomposing the sum (5), we have

$$|A|^2 = \sum_\lambda |A_\lambda| = \sum_{j=1}^{\lceil \log |A| \rceil} \sum_{\lambda \in S_{2^{j-1}}} |A_\lambda|.$$

Applying the pigeonhole principle, we deduce that there is some τ such that

$$\sum_{\lambda \in S_\tau} |A_\lambda| \geq \frac{|A|^2}{\lceil \log |A| \rceil} \geq \frac{|A|^2}{2 \log |A|}. \tag{6}$$

Since $\tau \leq |A|$, this implies that

$$|S_\tau| \geq \frac{|A|}{2 \log |A|}. \tag{7}$$

Also, since $|A_\lambda| < 2\tau$ for any $\lambda \in S_\tau$, we have

$$\tau |S_\tau| \gg \frac{|A|^2}{\log |A|}. \tag{8}$$

3.1 A lower bound for τ

Suppose[3] that $|A/A| \geq |A|^{\frac{4}{3}+\frac{2}{39}}$. Then, by Corollary 7,

$$|A/A + A| \gg |A/A|^{\frac{3}{4}}|A|^{\frac{1}{2}} \gg |A|^{\frac{3}{2}+\frac{1}{26}},$$

as required. Therefore, we may assume that $|A/A| \leq |A|^{\frac{4}{3}+\frac{2}{39}}$. In particular, by (8),

$$\tau|A|^{\frac{4}{3}+\frac{2}{39}} \geq \tau|A/A| \geq \tau|S_\tau| \gg \frac{|A|^2}{\log|A|}.$$

Therefore

$$\tau \gg \frac{|A|^{\frac{2}{3}-\frac{2}{39}}}{\log|A|}. \tag{9}$$

3.2 An upper bound for $d(A)$

Define P to be the subset of $A \times A$ lying on the union of the lines through the origin containing between τ and 2τ points. That is, $P = \cup_{\lambda \in S_\tau} \mathcal{A}_\lambda$. We will study vector sums coming from this point set by two different methods, and then compare the bounds in order to prove the theorem. To begin with, we use the methods from the paper [8] to obtain an upper bound for $d(A)$. The deduction of the forthcoming bound (11) is a minor variation of the first part of the proof of [8, Theorem 13].

After carrying out the aforementioned pigeonholing argument, we have a set of $|S_\tau|$ lines through the origin, each containing approximately τ points from $A \times A$. Label the lines $l_1, l_2, \ldots, l_{|S_\tau|}$ in increasing order of steepness. The line l_i has equation $y = q_i x$ and so $q_1 < q_2 < \cdots < q_{|S_\tau|}$. For any $1 \leq i \leq |S_\tau| - 1$, consider the sum set

$$\mathcal{A}_{q_i} + \mathcal{A}_{q_{i+1}} \cdot \Delta(A^{-1}) \subset (A + A/A) \times (A + A/A), \tag{10}$$

where $\Delta(B) = \{(b, b) : b \in B\}$. Note that $\mathcal{A}_{q_{i+1}} \cdot \Delta(A^{-1})$ has cardinality $|A_{q_{i+1}}A^{-1}|$, and therefore the set in (10) has at least $|A_{q_{i+1}}A^{-1}||A_{q_i}|$ elements, all of which lie in between l_i and l_{i+1}. This is a consequence of the observation of Solymosi that the sum set of m points on one line through the origin and n points on another line through the origin consists of mn points lying in between the two lines. It is important to note that this fact is dependent on the points lying inside the positive quadrant of the plane, which is why the assumption that A consists of strictly positive reals is needed for this proof.

[3] In order to simplify the some forthcoming calculations, we are a little careless with the logarithmic factors here. By making a weaker assumption that $|A/A| \geq |A|^{\frac{4}{3}+\frac{2}{39}}/(\log|A|)^C$, for an optimal choice of C, we can obtain a slightly smaller power of $\log|A|$ in the statement of Theorem 3.

Summing over all $1 \leq i < |S_\tau|$, applying the definition of $d(A)$ and using the bounds (8) and (6), we obtain

$$\begin{aligned}|A/A+A|^2 &\geq \sum_{i=1}^{|S_\tau|-1} |A_{q_i}||A/A_{q_{i+1}}| \\ &\geq |A|^{1/2} d^{1/2}(A) \sum_{i=1}^{|S_\tau|-1} |A_{q_i}||A_{q_{i+1}}|^{1/2} \\ &\gg \frac{|A|^{3/2} d^{1/2}(A)}{|S_\tau|^{1/2} \log^{1/2}|A|} \sum_{i=1}^{|S_\tau|-1} |A_{q_i}| \\ &\gg \frac{|A|^{7/2} d^{1/2}(A)}{|S_\tau|^{1/2} \log^{3/2}|A|}.\end{aligned}$$

This can be rearranged to obtain

$$d(A) \ll \frac{|A/A+A|^4 |S_\tau| \log^3|A|}{|A|^7}. \tag{11}$$

This bound will be utilised later in the proof. We now analyse the vector sums in a different way, based on the approach of [3].

3.3 Clustering setup

For each $\lambda \in S_\tau$, we identify an element from $\mathcal{A}_\lambda$, which we label $(a_\lambda, \lambda a_\lambda)$. These fixed points will have to be chosen with a little care later, but for the next part of the argument, we can think of the choice of $(a_\lambda, \lambda a_\lambda)$ as completely arbitrary, since the required bound holds whichever choice we make for these fixed points.

Then, fixing two distinct slopes λ and λ' from S_τ and following the observation of Balog [1], we note that at least $\tau|A|$ distinct elements of $(A/A+A) \times (A/A+A)$ are obtained by summing points from the two lines. Indeed,

$$\mathcal{A}_\lambda + (a_{\lambda'}, \lambda' a_{\lambda'}) \cdot \Delta(A^{-1}) \subset (A/A+A) \times (A/A+A).$$

Once again, these vector sums are all distinct and have slope in between λ and λ'.

Following the strategy of Konyagin and Shkredov [3], we split the family of $|S_\tau|$ slopes into clusters of $2M$ consecutive slopes, where $2 \leq 2M \leq |S_\tau|$ and M is a parameter to be specified later. For example, the first cluster is $U_1 = \{l_1, \ldots, l_{2M}\}$, the second is $U_2 = \{l_{2M+1}, \ldots, l_{4M}\}$, and so on. We then split each cluster arbitrarily into two disjoint subclusters of size M. For example, we have $U_1 = V_1 \sqcup W_1$ where $V_1 = \{l_1, \ldots, l_M\}$ and $W_1 = \{l_{M+1}, \ldots, l_{2M}\}$.

The idea is to show that each cluster determines many different elements of $(A + A/A) \times (A + A/A)$. Since the slopes of these elements are in between the maximal and minimal values in that cluster, we can then sum over all clusters without overcounting.

If a cluster contains exactly $2M$ lines, then it is called a *full cluster*. Note that there are $\left\lfloor \frac{|S_\tau|}{2M} \right\rfloor \geq \frac{|S_\tau|}{4M}$ full clusters, since we place exactly $2M$ lines in each cluster, with the possible exception of the last cluster which contains at most $2M$ lines.

The proceeding analysis will work in exactly the same way for any full cluster, and so for simplicity of notation we deal only with the first cluster U_1. We further simplify this by writing $U_1 = U$, $V_1 = V$ and $W_1 = W$.

Let μ denote the number of elements of $(A/A+A)\times(A/A+A)$ which lie in between l_1 and l_{2M}. Then[4]

$$\mu \geq \tau|A|M^2 - \sum_{\lambda_1,\lambda_3\in V,\lambda_2,\lambda_4\in W:\{\lambda_1,\lambda_2\}\neq\{\lambda_3,\lambda_4\}} \mathcal{E}(\lambda_1,\lambda_2,\lambda_3,\lambda_4), \tag{12}$$

where

$$\mathcal{E}(\lambda_1,\lambda_2,\lambda_3,\lambda_4) := |\{z\in(\mathcal{A}_{\lambda_1}+(a_{\lambda_2},\lambda_2 a_{\lambda_2})\cdot\Delta(A^{-1}))\cap(\mathcal{A}_{\lambda_3}+(a_{\lambda_4},\lambda_4 a_{\lambda_4})\cdot\Delta(A^{-1}))\}|.$$

In (12), the first term is obtained by counting sums from all pairs of lines in $V\times W$. The second error term covers the overcounting of elements that are counted more than once in the first term.

The next task is to obtain an upper bound for $\mathcal{E}(\lambda_1,\lambda_2,\lambda_3,\lambda_4)$ for an arbitrary quadruple $(\lambda_1,\lambda_2,\lambda_3,\lambda_4)$ which satisfies the aforementioned conditions.

Suppose that

$$z=(z_1,z_2)\in(\mathcal{A}_{\lambda_1}+(a_{\lambda_2},\lambda_2 a_{\lambda_2})\cdot\Delta(A^{-1}))\cap(\mathcal{A}_{\lambda_3}+(a_{\lambda_4},\lambda_4 a_{\lambda_4})\cdot\Delta(A^{-1})).$$

Then

$$\begin{aligned}(z_1,z_2) &= (a_1,\lambda_1 a_1)+(a_{\lambda_2}a^{-1},\lambda_2 a_{\lambda_2}a^{-1})\\ &= (a_3,\lambda_3 a_3)+(a_{\lambda_4}b^{-1},\lambda_4 a_{\lambda_4}b^{-1}),\end{aligned}$$

for some $a_1\in A_{\lambda_1}$, $a_3\in A_{\lambda_3}$ and $a,b\in A$. Therefore,

$$\begin{aligned}z_1 &= a_1+a_{\lambda_2}a^{-1}=a_3+a_{\lambda_4}b^{-1}\\ z_2 &= \lambda_1 a_1+\lambda_2 a_{\lambda_2}a^{-1}=\lambda_3 a_3+\lambda_4 a_{\lambda_4}b^{-1}\end{aligned}$$

3.4 Bounding $\mathcal{E}(\lambda_1,\lambda_2,\lambda_3,\lambda_4)$ in the case when $\lambda_4\neq\lambda_2$

Let us assume first that $\lambda_4\neq\lambda_2$. Note that this assumption implies that $\lambda_4\neq\lambda_1,\lambda_2,\lambda_3$. We have

$$0=\lambda_1 a_1+\lambda_2 a_{\lambda_2}a^{-1}-\lambda_3 a_3-\lambda_4 a_{\lambda_4}b^{-1}-\lambda_4(a_1+a_{\lambda_2}a^{-1}-a_3-a_{\lambda_4}b^{-1}),$$

and thus

$$0=a_{\lambda_2}(\lambda_2-\lambda_4)a^{-1}+(\lambda_1-\lambda_4)a_1+(\lambda_4-\lambda_3)a_3. \tag{13}$$

Note that the values $\lambda_1-\lambda_4, a_{\lambda_2}(\lambda_2-\lambda_4)$ and $\lambda_4-\lambda_3$ are all non-zero. We have shown that each contribution to $\mathcal{E}(\lambda_1,\lambda_2,\lambda_3,\lambda_4)$ determines a solution to (13) with $(a,a_1,a_3)\in A\times A_{\lambda_1}\times A_{\lambda_3}$. Furthermore, the solution to (13) that we obtain via this deduction is unique, and so a bound for $\mathcal{E}(\lambda_1,\lambda_2,\lambda_3,\lambda_4)$ will follow from a bound to the number of solutions to (13).

It therefore follows from an application of Lemma 5 that

$$\mathcal{E}(\lambda_1,\lambda_2,\lambda_3,\lambda_4)\leq C\cdot d(A^{-1})^{1/3}|A|^{1/3}(A)\tau^{4/3}=C\cdot d(A)^{1/3}|A|^{1/3}(A)\tau^{4/3},$$

[4] For the sake of simplicity of presentation, a small abuse of notation is made here. The lines in V and W are identified with their slopes. In this way, the notation $\lambda_i\in V$ is used as a shorthand for $\{(x,y):y=\lambda_i x\}\in V$.

where C is an absolute constant. Therefore,

$$\mu \geq M^2|A|\tau - M^4 C d^{1/3}(A)|A|^{1/3}\tau^{4/3} - \sum_{\lambda_1,\lambda_3 \in V, \lambda_2 \in W : \lambda_1 \neq \lambda_3} \mathcal{E}(\lambda_1, \lambda_2, \lambda_3, \lambda_2) \tag{14}$$

We now impose a condition on the parameter M (recall that we will choose an optimal value of M at the conclusion of the proof) to ensure that the first error term is dominated by the main term. We need

$$CM^4 d^{1/3}(A)|A|^{1/3}\tau^{4/3} \leq \frac{M^2|A|\tau}{2},$$

which simplifies to

$$M \leq \frac{|A|^{1/3}}{\sqrt{2C}d^{1/6}(A)\tau^{1/6}}. \tag{15}$$

With this restriction on M, we now have

$$\mu \geq \frac{M^2|A|\tau}{2} - \sum_{\lambda_1,\lambda_3 \in V, \lambda_2 \in W : \lambda_1 \neq \lambda_3} \mathcal{E}(\lambda_1, \lambda_2, \lambda_3, \lambda_2). \tag{16}$$

It remains to bound this second error term.

3.5 Bounding $\mathcal{E}(\lambda_1, \lambda_2, \lambda_3, \lambda_4)$ in the case $\lambda_4 = \lambda_2$

It is in this case that we need to take care to make good choices for the fixed points $(a_\lambda, \lambda a_\lambda)$ on each line l_λ.

Fix $\lambda_2 \in W$. We want to prove that there is a choice for $(a_{\lambda_2}, \lambda a_{\lambda_2}) \in \mathcal{A}_{\lambda_2}$ such that

$$\sum_{\lambda_1,\lambda_3 \in V : \lambda_1 \neq \lambda_3} \mathcal{E}(\lambda_1, \lambda_2, \lambda_3, \lambda_2) \ll M^2\tau^{1/3}|A|^{4/3}.$$

We will do this using the Szemerédi-Trotter Theorem. Consider the sum

$$\sum_{a_{\lambda_2} \in A_{\lambda_2}} \sum_{\lambda_1,\lambda_3 \in V : \lambda_1 \neq \lambda_3} |\{z \in (\mathcal{A}_{\lambda_1} + (a_{\lambda_2}, \lambda_2 a_{\lambda_2}) \cdot \Delta(A^{-1})) \cap (\mathcal{A}_{\lambda_3} + (a_{\lambda_2}, \lambda_2 a_{\lambda_2}) \cdot \Delta(A^{-1}))\}|.$$

Suppose that

$$z = (z_1, z_2) \in (\mathcal{A}_{\lambda_1} + (a_{\lambda_2}, \lambda_2 a_{\lambda_2}) \cdot \Delta(A^{-1})) \cap (\mathcal{A}_{\lambda_3} + (a_{\lambda_2}, \lambda_2 a_{\lambda_2}) \cdot \Delta(A^{-1})).$$

Then

$$\begin{aligned}(z_1, z_2) &= (a_1, \lambda_1 a_1) + (a_{\lambda_2} a^{-1}, \lambda_2 a_{\lambda_2} a^{-1}) \\ &= (a_3, \lambda_3 a_3) + (a_{\lambda_2} b^{-1}, \lambda_2 a_{\lambda_2} b^{-1}),\end{aligned}$$

for some $a_1 \in A_{\lambda_1}$, $a_3 \in A_{\lambda_3}$ and $a, b \in A$. Therefore,

$$\begin{aligned}z_1 &= a_1 + a_{\lambda_2} a^{-1} = a_3 + a_{\lambda_2} b^{-1} \\ z_2 &= \lambda_1 a_1 + \lambda_2 a_{\lambda_2} a^{-1} = \lambda_3 a_3 + \lambda_2 a_{\lambda_2} b^{-1}.\end{aligned}$$

We have

$$0 = \lambda_1 a_1 + \lambda_2 a_{\lambda_2} a^{-1} - \lambda_3 a_3 - \lambda_2 a_{\lambda_2} b^{-1} - \lambda_1(a_1 + a_{\lambda_2} a^{-1} - a_3 - a_{\lambda_2} b^{-1}),$$

and thus

$$\frac{\lambda_3-\lambda_1}{\lambda_2-\lambda_1}a_3 = a_{\lambda_2}(a^{-1}-b^{-1}). \tag{17}$$

As in the previous subsection, this shows that the quantity

$$\sum_{a_{\lambda_2}\in A_{\lambda_2}} \sum_{\lambda_1\neq\lambda_3\in V} |(\mathcal{A}_{\lambda_1}+(a_{\lambda_2},\lambda_2 a_{\lambda_2})\cdot\Delta(A^{-1}))\cap(\mathcal{A}_{\lambda_3}+(a_{\lambda_2},\lambda_2 a_{\lambda_2})\cdot\Delta(A^{-1}))|$$

is no greater than the number of solutions to (17) such that $(\lambda_1,\lambda_3,a,b,a_{\lambda_2},a_3)\in V\times V\times A\times A\times A_{\lambda_2}\times A_{\lambda_3}$.

Fix, $\lambda_1,\lambda_3\in V$ such that $\lambda_1\neq\lambda_3$. Let $Q=A^{-1}\times A_{\lambda_3}$. Define $l_{m,c}$ to be the line with equation $\frac{\lambda_3-\lambda_1}{\lambda_2-\lambda_1}y=m(x-c)$ and define L to be the set of lines

$$L=\{l_{a_{\lambda_2},b^{-1}} : a_{\lambda_2}\in A_{\lambda_2}, b\in A\}.$$

Note that $|Q|\approx|L|\approx\tau|A|$ and so

$$I(Q,L)\ll(\tau|A|)^{4/3}.$$

Repeating this analysis via the Szemerédi-Trotter Theorem for each pair of distinct $\lambda_1,\lambda_3\in V$, it follows that the number of solutions to (17) is $O(M^2(\tau|A|)^{4/3})$. In summary,

$$\sum_{a_{\lambda_2}\in A_{\lambda_2}} \sum_{\lambda_1\neq\lambda_3\in V} |(\mathcal{A}_{\lambda_1}+(a_{\lambda_2},\lambda_2 a_{\lambda_2})\Delta(A^{-1}))\cap(\mathcal{A}_{\lambda_3}+(a_{\lambda_2},\lambda_2 a_{\lambda_2})\Delta(A^{-1}))|\ll M^2(\tau|A|)^{4/3}.$$

Therefore, by the pigeonhole principle, there is some $a_{\lambda_2}\in A_{\lambda_2}$ such that

$$\sum_{\lambda_1\neq\lambda_3\in V} |(\mathcal{A}_{\lambda_1}+(a_{\lambda_2},\lambda_2 a_{\lambda_2})\cdot\Delta(A^{-1}))\cap(\mathcal{A}_{\lambda_3}+(a_{\lambda_2},\lambda_2 a_{\lambda_2})\cdot\Delta(A^{-1}))|\ll M^2\tau^{1/3}|A|^{4/3}. \tag{18}$$

We can then choose the fixed point $(a_{\lambda_2},\lambda_2 a_{\lambda_2})$ on l_{λ_2} to be that corresponding to the value a_{λ_2} satisfying inequality (18). This in fact shows that

$$\sum_{\lambda_1\neq\lambda_3\in V}\mathcal{E}(\lambda_1,\lambda_2,\lambda_3,\lambda_2)\ll M^2\tau^{1/3}|A|^{4/3}. \tag{19}$$

We repeat this process for each $\lambda_2\in W$ to choose a fix point for each line with slope in W. Summing over all $\lambda_2\in W$, we now have

$$\sum_{\lambda_1\neq\lambda_3\in V,\lambda_2\in W}\mathcal{E}(\lambda_1,\lambda_2,\lambda_3,\lambda_2)\ll M^3\tau^{1/3}|A|^{4/3}. \tag{20}$$

We have a bound for error term in (16). Still, we need to impose a condition on M so that this error term is dominated by the main term. We need

$$M^3\tau^{1/3}|A|^{4/3}\leq\frac{M^2|A|\tau}{4},$$

which simplifies to

$$M\leq\frac{\tau^{2/3}}{4|A|^{1/3}}. \tag{21}$$

With this restriction on M, we now have

$$\mu\geq\frac{M^2|A|\tau}{4}. \tag{22}$$

Our integer parameter M must satisfy (15) and (21). We therefore choose

$$M := \min\left\{\left\lfloor \frac{|A|^{1/3}}{\sqrt{2C}d^{1/6}(A)\tau^{1/6}}\right\rfloor, \left\lfloor \frac{\tau^{2/3}}{4|A|^{1/3}}\right\rfloor\right\}.$$

Summing over the full clusters, of which there are at least $\frac{|S_\tau|}{4M}$, yields

$$|A/A+A|^2 \geq \frac{|S_\tau|}{4M}\frac{M^2}{4}|A|\tau \tag{23}$$

$$\gg |S_\tau|M|A|\tau \tag{24}$$

3.6 Choosing M - case 1

Suppose first that $M = \left\lfloor \frac{|A|^{1/3}}{\sqrt{2C}d^{1/6}(A)\tau^{1/6}}\right\rfloor$.

Recall that we need $2 \leq 2M \leq |S_\tau|$. It is easy to check that the upper bound for M is satisfied. Indeed,

$$2M \leq \frac{2}{\sqrt{2C}}|A|^{1/3} \leq \frac{|A|}{2\log|A|} \leq |S_\tau|.$$

The first inequality above uses the fact that $d(A) \geq 1$ for all A (since one can take C to be a singleton in the (4)), as well as the bound $\tau \geq 1$. The second inequality is true for sufficiently large $|A|$, and the third is (7). Since smaller sets can be dealt with by choosing sufficiently small implied constants in the statement, we may assume that $2M \leq |S_\tau|$.

Assume first that $M \geq 1$ (we will deal with the other case later). Then, by (24) and the definition of M

$$|A/A+A|^2 \gg \frac{|S_\tau||A|^{4/3}\tau^{5/6}}{d^{1/6}(A)}.$$

Applying the inequality $|S_\tau|\tau \gg \frac{|A|^2}{\log|A|}$, it follows that

$$d^{1/6}(A)|A/A+A|^2 \gg \frac{|A|^3|S_\tau|^{1/6}}{\log^{5/6}|A|}. \tag{25}$$

After bounding the left hand side of this inequality using (11), we obtain

$$\frac{|A/A+A|^{2/3}|S_\tau|^{1/6}\log^{1/2}|A|}{|A|^{7/6}}|A/A+A|^2 \gg d^{1/6}(A)|A/A+A|^2 \gg \frac{|A|^3|S_\tau|^{1/6}}{\log^{5/6}|A|}.$$

Rearranging this expression leads to the bound

$$|A/A+A| \gg \frac{|A|^{25/16}}{\log^{1/2}|A|},$$

which is stronger than the claim of the theorem.

It remains is to consider what happens if $M \leq 1$. Indeed, if this is the case, then

$$\frac{|A|^{1/3}}{\sqrt{8C}d^{1/6}(A)\tau^{1/6}} < 1$$

and so

$$\frac{|A|^{1/3}}{d^{1/6}(A)\tau^{1/6}} \ll 1.$$

After applying the bound $\tau \leq |A|^2/|S_\tau|$, it follows that

$$\frac{|S_\tau|^{1/6}}{d^{1/6}(A)} \ll 1 \ll \frac{|A/A+A|^2}{|A|^3},$$

where the latter inequality is a consequence of Theorem 1. In particular, this implies that (25) holds. We can then repeat the earlier analysis and once again reach the conclusion that

$$|A/A+A| \gg \frac{|A|^{\frac{3}{2}+\frac{1}{16}}}{\log^{1/2}|A|}.$$

3.7 Choosing M - case 2

Suppose now that $M = \left\lfloor \frac{\tau^{2/3}}{4|A|^{1/3}} \right\rfloor$.

Again, we need to check that $2 \leq 2M \leq |S_\tau|$. If the lower bound does not hold then (9) gives a contradiction for sufficiently large $|A|$. Smaller sets can be dealt with by choosing sufficiently small implied constants in the statement. If the upper bound does not hold then

$$\frac{\tau^{2/3}}{|A|^{1/3}} \geq 2M > |S_\tau|.$$

Multiplying both sides of this inequality by τ and applying (8) gives the contradiction

$$|A|^{5/3} \geq \tau^{5/3} \gg \frac{|A|^{7/3}}{\log|A|}.$$

Since this choice of M is valid, we can now conclude the proof. From (9), we have

$$M \gg \frac{\tau^{2/3}}{|A|^{1/3}} \gg \frac{|A|^{\frac{1}{13}}}{\log^{\frac{2}{3}}|A|}.$$

Then, by (24) and (8),

$$|A/A+A|^2 \gg \frac{|A|^3}{\log|A|}M \gg \frac{|A|^{3+\frac{1}{13}}}{\log^{5/3}|A|}.$$

We conclude that

$$|A/A+A| \gg \frac{|A|^{\frac{3}{2}+\frac{1}{26}}}{\log^{5/6}|A|},$$

and so the proof is complete. ◀

References

1. A. Balog 'A note on sum-product estimates', *Publ. Math. Debrecen* **79**, no. 3-4 (2011), 283-289.
2. G. Elekes, M. Nathanson and I. Ruzsa, 'Convexity and sumsets', *J. Number Theory.* **83** (1999), 194-201.
3. S. Konyagin and I. Shkredov, 'On sum sets of sets, having small product set', *Proc. Steklov Inst. Math.* **290** (2015), 288-299.
4. S. Konyagin and I. Shkredov, 'New results on sums and products in $\mathbb{R}$', *Proc. Steklov Inst. Math.* **294** (2016), 87-98.

5 O. Roche-Newton, I. Z. Ruzsa, C.-Y. Shen and I. D. Shkredov 'On the size of the set $AA + A$', *arxiv:1801.10431* (2018).

6 M. Rudnev, I. D. Shkredov and S. Stevens, 'On the energy variant of the sum-product conjecture', *arxiv:1607.05053* (2016).

7 G. Shakan 'On higher energy decompositions and the sum-product phenomenon', *arXiv:1803.04637* (2018).

8 I. Shkredov, 'On a question of A. Balog', *Pacific J. Math.* 280 (2016), no. 1, 227-240.

9 J. Solymosi, 'Bounding multiplicative energy by the sumset', *Adv. Math.* **222** (2009), 402-408.

10 T. Tao, V. Vu. 'Additive combinatorics' *Cambridge University Press* (2006).

Fractal Dimension and Lower Bounds for Geometric Problems

Anastasios Sidiropoulos
Department of Computer Science, University of Illinois at Chicago
Chicago IL, USA
sidiropo@uic.edu

Kritika Singhal
Department of Mathematics, The Ohio State University
Columbus OH, USA
singhal.53@osu.edu

Vijay Sridhar
Department of Computer Science and Engineering, The Ohio State University
Columbus OH, USA
sridhar.38@osu.edu

Abstract

We study the complexity of geometric problems on spaces of low *fractal dimension*. It was recently shown by [Sidiropoulos & Sridhar, SoCG 2017] that several problems admit improved solutions when the input is a pointset in Euclidean space with fractal dimension smaller than the ambient dimension. In this paper we prove nearly-matching lower bounds, thus establishing nearly-optimal bounds for various problems as a function of the fractal dimension.

More specifically, we show that for any set of n points in d-dimensional Euclidean space, of fractal dimension $\delta \in (1, d)$, for any $\varepsilon > 0$ and $c \geq 1$, any c-spanner must have treewidth at least $\Omega\left(\frac{n^{1-1/(\delta-\epsilon)}}{c^{d-1}}\right)$, matching the previous upper bound. The construction used to prove this lower bound on the treewidth of spanners, can also be used to derive lower bounds on the running time of algorithms for various problems, assuming the Exponential Time Hypothesis. We provide two prototypical results of this type:

- For any $\delta \in (1, d)$ and any $\varepsilon > 0$, d-dimensional Euclidean TSP on n points with fractal dimension at most δ cannot be solved in time $2^{O\left(n^{1-1/(\delta-\varepsilon)}\right)}$. The best-known upper bound is $2^{O(n^{1-1/\delta}\log n)}$.
- For any $\delta \in (1, d)$ and any $\varepsilon > 0$, the problem of finding k-pairwise non-intersecting d-dimensional unit balls/axis parallel unit cubes with centers having fractal dimension at most δ cannot be solved in time $f(k)n^{O\left(k^{1-1/(\delta-\varepsilon)}\right)}$ for any computable function f. The best-known upper bound is $n^{O(k^{1-1/\delta}\log n)}$.

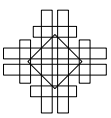
The above results nearly match previously known upper bounds from [Sidiropoulos & Sridhar, SoCG 2017], and generalize analogous lower bounds for the case of ambient dimension due to [Marx & Sidiropoulos, SoCG 2014].

2012 ACM Subject Classification Theory of computation → Lower bounds and information complexity, Theory of computation → Computational geometry

Keywords and phrases fractal dimension, treewidth, spanners, lower bounds, exponential time hypothesis

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.70

Related Version A full version of the paper is available at `https://arxiv.org/abs/1712.04595`.



Funding Supported by NSF under award CAREER 1453472 and grant CCF 1423230.

1 Introduction

The *curse of dimensionality* is a general phenomenon in computational geometry, asserting that the complexity of many problems increases rapidly with the dimension of the input. Sets of fractional dimension can be used to model various processes and phenomena in science and engineering [25]. Recently, the complexity of various geometric optimization problems was studied as a function of the fractal dimension of the input [23]. It was shown that, for several problems, improved algorithms can be obtained when the fractal dimension is smaller than the ambient dimension.

Interestingly, the algorithms obtained in [23] nearly match the best-possible algorithms for integral dimension. In this paper, we give nearly-matching lower bounds, assuming the Exponential Time Hypothesis (ETH). We remark that there are several different definitions of fractal dimension that can be considered. Our results indicate that, for the case of Euclidean pointsets, the definition of fractal dimension we consider is the "correct" one for certain computational problems. That is, it precisely generalizes the dependence of the running time on the ambient dimension.

1.1 Our contribution

We obtain nearly-optimal lower bounds for various prototypical geometric problems. Our results are obtained via a general method that could be applicable to other problems.

Spanners

We begin with a lower bound on the treewidth of spanners. It is known that any set of n points in $\mathbb{R}^d$ admits a $(1+\varepsilon)$-spanner of size $n(1/\varepsilon)^{O(d)}$ [22, 27]. This result has been generalized for the case of fractal dimension. Specifically, it was shown in [23] that any n-point set in $O(1)$-dimensional Euclidean space, of fractal dimension $\delta > 1$, admits a $(1+\varepsilon)$-spanner of size $n(1/\varepsilon)^{O(d)}$, and of pathwidth $O(n^{1-1/\delta}\log n)$. We show the following lower bound, which establishes that the upper bound from [23] is essentially best-possible.

▶ **Theorem 1.** *Let $d \geq 2$ be an integer. Then for all $\delta \in (1, d)$, for all $\varepsilon > 0$ and for all $n_0 \in \mathbb{N}$, there exists a set of $n \geq n_0$ points $P \subset \mathbb{R}^d$, of fractal dimension at most δ', where $|\delta - \delta'| \leq \varepsilon$, such that for any $c \geq 1$, any c-spanner G of P has $tw(G) = \Omega\left(\frac{n^{1-1/\delta'}}{c^{d-1}}\right)$.*

Independent Set of Unit Balls

We consider the k-Independent Set of Unit Balls in $\mathbb{R}^d$, which is a prototypical geometric optimization problem, parameterized by the optimum. In this problem given a set of n unit balls in $\mathbb{R}^d$, we seek to find a set of k pairwise non-intersecting balls. It is known that this problem can be solved in time $n^{O(k^{1-1/d})}$, for any $d \geq 2$ [1, 19], and that there is no algorithm with running time $f(k)n^{o(k^{1-1/d})}$, for any computable function f, assuming ETH [19] (see also [17]). The upper bound has been generalized for fractal dimension as follows: It has been shown that when the set of centers of the balls has fractal dimension δ, the problem can be solved in time $n^{O(k^{1-1/\delta}\log n)}$ [23]. We show the following lower bound on the running time, which nearly matches this upper bound, up to a logarithmic term.

▶ **Theorem 2.** *Let $d \geq 2$ be an integer, and let $1 < \delta' < \delta < d$. If for all k, and for some computable function f, there exists an $f(k)n^{o(k^{1-1/\delta'})}$ time algorithm for finding k pairwise non-intersecting open balls in a collection of n unit balls with the centers having fractal dimension at most δ, then ETH fails.*

Euclidean TSP

It is known that TSP on a set of n points in d-dimensional Euclidean space can be solved in time $2^{1-1/d}n^{O(1)}$ [24], and that there is no algorithm with running time $2^{O(n^{1-1/d-\varepsilon})}$, for any $\varepsilon > 0$, assuming ETH. The upper bound has been generalized to the case of fractal dimension as follows. It has been shown that for a set of fractal dimension $\delta > 1$, in $O(1)$-dimensional Euclidean space, TSP can be solved in time $2^{O(n^{1-1/\delta}\log n)}$. Here, we obtain the following nearly-tight lower bound.

▶ **Theorem 3.** *Let $d \geq 2$ be an integer. Then, for all $\delta \in (2, d)$, for all $\delta' < \delta$ and for all $n_0 \in \mathbb{N}$, if there exists $n \geq n_0$ such that Euclidean TSP in $\mathbb{R}^d$ on all pointsets of size n and fractal dimension at most δ can be solved in time $2^{O(n^{1-1/\delta'})}$, then ETH fails.*

1.2 Overview of techniques

We now briefly highlight the main technical tools used in the paper.

High-level idea

We derive our lower bounds by adapting a method from the case of graph problems. It is known that, for many problems on graphs, large treewidth implies large running time lower bounds (see, e.g. [18]). This is typically done by exploiting the duality between treewidth and grid minors. Specifically, it is known that, for any r, graphs of treewidth at least $f(r)$, for some function f, have the $(r \times r)$-grid as a minor [21]. In fact, the function f is known to be linear for planar graphs [20], and polynomial in general [5]. One can often obtain a lower bound on the running time by using the grid minor to embed a large hard instance in the input. We apply the above approach to the geometric setting by relating fractal dimension to treewidth. Specifically, we construct a pointset in Euclidean space, such that any $O(1)$-spanner must have large treewidth.

From fractal dimension to treewidth

A main technical ingredient for obtaining nearly-optimal lower bounds is constructing pointsets such that the treewidth of any $O(1)$-spanner is as large as possible. For the case of exposition, we will describe the construction in the continuous case. We construct some $X \subset \mathbb{R}^d$, and we discretize X by taking some $O(1)$-approximate[1] ε-net N_ε of X. Let us refer to the fractal dimension of the resulting infinite family of nets N_ε, as the fractal dimension of X (see Section 1.4 for precise definitions). Our goal is to construct some X, with some fixed fractal dimension $\delta \in (1, d]$, such that the treewidth of any $O(1)$-spanner of N_ε is as large as possible as a function of $1/\varepsilon$.

[1] Recall that a $O(1)$-approximate r-net in some metric space (X, ρ) is some $N \subseteq X$, such that for all $x \neq y \in N$, $\rho(x, y) > r$, and for all $z \notin N$, $\rho(z, N) = \inf_{y \in N} d(z, y) = O(r)$.

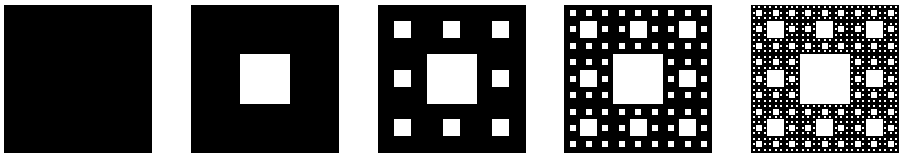


Figure 1 The first few iterations of the Sierpiński carpet.

Figure 2 The first few iterations of the Cantor ternary set.

A first failed attempt: the Sierpiński carpet

Let us now briefly describe the construction and point out the main technical challenges. A natural first attempt in $\mathbb{R}^2$ is to let X be the Sierpiński carpet. This is a set obtained from the unit square by removing the central square of side length $1/3$, and by recursing on the remaining 8 sub-squares (see Figure 1). Unfortunately, this construction does not lead to a tight treewidth lower bound. Specifically, the resulting set has fractal dimension $\delta = \log 8/\log 3$, while there exist $O(1)$-spanners of treewidth $O(n^{1-1/\gamma})$, where γ is a constant arbitrarily close to $\log 6/\log 3$.

Intuitively, this happens for the following reason. Let S_ε be any $O(1)$-spanner for N_ε. Then, the largest possible grid minor in S_ε does not use most of the vertices in S_ε. Thus, roughly speaking, we can obtain a larger grid minor by constructing a set X so that as few vertices of S_ε as possible are being "wasted".

Constructing a treewidth-extremal fractal: the Cantor crossbar

Using the above observation, we define the set X as follows. We first recall that the Cantor set $\mathcal{C}$ is obtained from the unit interval by removing the central interval of length $1/3$, and recursing on the other two (see Figure 2). We define $\mathcal{C}'$ to be the Cartesian product of $\mathcal{C}$ with $[0,1]$, and we set X to be the union of two copies of $\mathcal{C}'$, where one is rotated by $\pi/2$. We refer to the resulting set as the *Cantor crossbar* (see Figure 3). We can show that the resulting set achieves a nearly-optimal treewidth lower bound.

The above construction can be generalized to the case where the ambient dimension is $d \geq 2$ as follows. Recall that, for any $d' \geq 1$, the *Cantor dust* in $\mathbb{R}^{d'}$, denoted by $\mathcal{D}_{d'}$, is the Cartesian product of d' copies of the Cantor set (see Figure 4). Let $e_1, \ldots, e_d$ be the standard orthonormal basis in $\mathbb{R}^d$. For each $i \in \{1, \ldots, d\}$, we define $\mathcal{T}_i$ to be the Cartesian product of $\mathcal{D}_{d-1}$ with $[0,1]$, rotated so that $[0,1]$ is parallel to e_i. Finally, we set $X = \mathcal{T}_1 \cup \ldots \cup \mathcal{T}_d$ (see Figure 5).

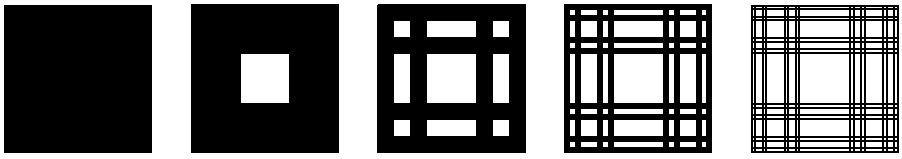


Figure 3 The first few iterations of the Cantor crossbar in $\mathbb{R}^2$.

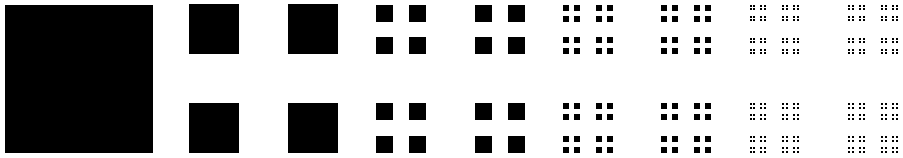


Figure 4 The first few iterations of the Cantor dust in $\mathbb{R}^2$.

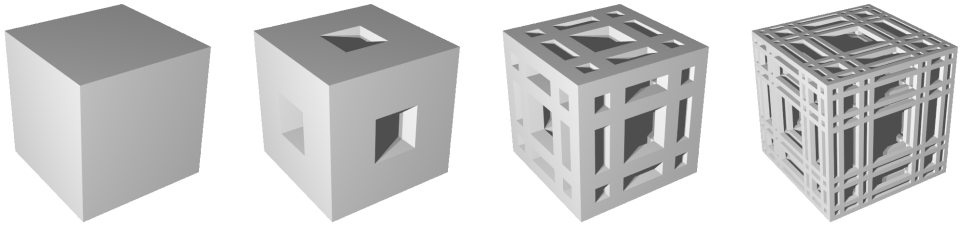


Figure 5 The first few iterations of the Cantor crossbar in $\mathbb{R}^3$.

The above construction gives a set with some fixed fractal dimension δ, for each fixed $d \geq 2$. We can generalize the construction so that δ attains any desired value in the range $(1, d]$. The only difference is that, when defining the Cantor dust, we start with a Cantor set of smaller dimension. This can be done by removing the central interval of length $\alpha \in (0, 1)$, instead of $1/3$, and recursing on the remaining two intervals of length $(1 - \alpha)/2$.

From spanner lower bounds to running time lower bounds

The above construction of the Cantor crossbar leads to a nearly-optimal lower bound for the treewidth of $O(1)$-spanners. We next use this construction to obtain running time lower bounds. Informally, a typical NP-hardness reduction for some geometric problem in the plane works as follows: One encodes some known computationally hard problem by constructing "gadgets" that are arranged in a grid-like fashion in $\mathbb{R}^2$ (see, e.g. [19]). More generally, for problems in $\mathbb{R}^d$, the gadgets are arranged along some d-dimensional grid. We follow a similar approach, with the main difference being that we arrange the gadgets along a Cantor crossbar.

1.3 Other related work

There has been a large body of work on determining the effect of *doubling* dimension on the complexity of various geometric problems [10, 2, 6, 14, 8, 16, 4, 3, 9, 26]. Other notions of dimension that have been considered include low-dimensional negatively curved spaces [15], growth-restricted metrics [12], as well as generalizations of doubling dimension to metrics of so-called bounded global growth [11]. In all of the above lines of research the goal is to extend tools and ideas from the Euclidean setting to more general geometries. In contrast, we study restricted classes of Euclidean instances, with the goal of obtaining better bounds than what is possible in the general case.

1.4 Preliminaries

We give some definitions that are used throughout the paper.

▶ **Definition 4** (ε-covering)**.** Let X be a set and $A \subseteq X$. Then, A is an ε-covering of X if for every $x \in X$, there exists $a \in A$ such that $\text{dist}(x, a) \leq \varepsilon$.

▶ **Definition 5** (ε-packing)**.** A set A is called an ε-packing if for all $x, y \in A$, $\text{dist}(x, y) \geq \varepsilon$.

▶ **Definition 6** (ε-net)**.** A set $A \subseteq X$ is called an ε-net of X if A is an ε-packing as well as an ε-covering of X.

▶ **Definition 7** (Fractal dimension)**.** [23] Given $x \in \mathbb{R}^d$ and $r > 0$, let $B(x, r)$ denote the ball of radius r centered at x. The *fractal dimension* of some $P \subseteq \mathbb{R}^d$, denoted by $\dim_{\mathsf{f}}(P)$, is defined as the infimum δ, such that for any $\varepsilon > 0$ and $r \geq 2\varepsilon$, for any ε-net N of P, and for any $x \in \mathbb{R}^d$, we have $|N \cap B(x, r)| = O((r/\varepsilon)^\delta)$.

We have the following lemmas showing invariance of fractal dimension under certain operations.

▶ **Lemma 8.** *Let $d \geq 1$ be an integer, let $0 < \delta \leq d$ and let $c > 0$ be some constant. Let $P \subset \mathbb{R}^d$ be a pointset such that $\dim_{\mathsf{f}}(P) = \delta$. Let P' be the pointset obtained by uniformly scaling the points of P about the origin by a factor of c. Then $\dim_{\mathsf{f}}(P') = \delta$.*

▶ **Lemma 9.** *Let $d \geq 1$ be some integer, let $0 < \delta \leq d$ and let $c > 0, k > 0$ be some constants. Let $P \subset \mathbb{R}^d$ be a pointset such that $\dim_{\mathsf{f}}(P) = \delta$ and for all $u, v \in P$, $d(u, v) > 4c$. For all $p \in P$ let $S_p \subset \mathbb{R}^d$ be a set of points such that $|S_p| \leq k$ and for all $x \in S_p$, $d(x, p) \leq c$. Let $P' = \bigcup_{p \in P} S_p$. Then $\dim_{\mathsf{f}}(P') = \delta$.*

▶ **Definition 10** (c-spanner)**.** For any pointset $P \subset \mathbf{R}^d$, and for any $c \geq 1$, a *c-spanner* for P is a graph G with $V(G) = P$, such that for all $x, y \in P$, we have

$$||x - y||_2 \leq d_G(x, y) \leq c \cdot ||x - y||_2,$$

where d_G denotes the shortest path distance in G.

▶ **Definition 11** (Treewidth)**.** [7] Let G be a graph, T a tree and $\mathcal{V} = \{V_t\}_{t \in T}$ be a family of vertex sets $V_t \subseteq V(G)$ indexed by the vertices t of T. The pair $(T, \mathcal{V})$ is called a tree-decomposition of G if it satisfies the following three conditions:

1. $V(G) = \cup_{t \in T} V_t$.
2. For every edge $e \in G$, there exists a $t \in T$ such that both ends of e lie in V_t.
3. $V_{t_1} \cap V_{t_3} \subseteq V_{t_2}$, whenever t_2 lies in the unique path joining t_1 and t_3 in T.

The width of $(T, \mathcal{V})$ is the number $\max\{|V_t| \ : \ t \in T\}$ and the *treewidth* of G is the least width of any tree-decomposition of G.

Exponential Time Hypothesis. The Exponential Time Hypothesis states that there is no $2^{o(n)}$ algorithm to solve the n-variable 3SAT.

1.5 Organization

This paper is organized as follows. In section 2, we present lower bounds on the treewidth of spanners for arbitrary pointsets with integral dimension, and with fractal dimension. In section 3, we present running time lower bound on the Independent Set of Balls problem on pointsets with arbitrary fractal dimension in $\mathbf{R}^d$. The proof of Theorem 3 can be found in the full version of the paper. This theorem proves running time lower bound on the Euclidean TSP problem on pointsets with arbitrary fractal dimension in $\mathbf{R}^d$.

2 Lower bounds on the treewidth of spanners

In this section, we obtain lower bounds on the treewidth of spanners for arbitrary pointsets. In subsection 2.1, we consider pointsets with integral fractal dimension. In subsection 2.2, we consider a discretized version of the Sierpiński carpet whose fractal dimension is less than two but greater than 1. In subsection 2.3, we use a carefully chosen inductive construction to obtain a specific fractal pointset of fractal dimension $\frac{\log 6}{\log 3}$. This pointset gives us a nearly tight lower bound on the treewidth of a spanner. We finally generalize this construction in subsection 2.4 and present the proof of Theorem 1.

2.1 Treewidth and integral dimension

We obtain lower bounds on the treewidth of spanners for pointsets with integral fractal dimension. We will make use of the following theorem due to Kozawa *et al.*[13] for the proofs in this section.

▶ **Theorem 12** ([13]). *The treewidth of the d-dimensional grid on n vertices is $\Theta(n^{1-1/d})$.*

▶ **Theorem 13.** *For any integer $d \geq 1$, there exists a set of n points $P \subseteq \mathbb{R}^d$ such that for any $c \geq 1$, and for any c-spanner G of P, $\mathsf{tw}(G) = \Omega\left(\frac{n^{1-1/d}}{c^{d-1}}\right)$.*

Proof. Fix an integer $d \geq 1$. Let $n \in \mathbf{N}$ be such that $n^{\frac{1}{d}}$ is an integer. Let $P = \{1, 2, \cdots, n^{\frac{1}{d}}\}^d$ and let G be any c-spanner of P. Let $p = (p_1, \ldots, p_d)$ be a point in P. We define P' and X' as follows:

$$P' = \{p \in P \mid \exists\ 1 \leq i \leq d \text{ such that } \forall\ j \neq i,\ (p_j \bmod (c+1)) = 1\}.$$

$$X = \{p \in P' \mid \forall\ 1 \leq i \leq d,\ (p_i \bmod (c+1)) = 1\}.$$

Consider the points in P'. We call a row of points in P' that is parallel to one of the d axes a *full row* if this row consists of exactly $n^{\frac{1}{d}}$ points with adjacent points unit distance apart. Consider for any $1 \leq i \leq d$, the points of any pair of full rows R and T that are both parallel to the i^{th} axis. We have that for any pair of consecutive points x_1, x_2 in R and for any pair of consecutive points y_1, y_2 in T, no shortest path in G joining x_1 and x_2 can intersect any shortest path in G joining y_1 and y_2. Suppose not, then let z be a point of intersection between two such shortest paths. Since G is a c-spanner and consecutive points in any row are distance 1 apart, we have $d_G(x_1, z) + d_G(x_2, z) \leq c$ and similarly, $d_G(y_1, z) + d_G(y_2, z) \leq c$. But this implies that at least one of $d_G(x_1, y_1)$, $d_G(x_1, y_2)$, $d_G(x_2, y_1)$ and $d_G(x_2, y_2)$ is at most c due to triangle inequality. This is a contradiction because G is non-contracting and the distance between R and T is at least $c+1$ by our choice of R and T.

Now if we consider a shortest path between every pair of consecutive points in the row R and concatenate these paths, remove all loops, then we can obtain a path from one end of this row to the other end. Doing the same for all full rows in P', we end up with a set of paths traversing the points in the full rows. Moreover from the earlier argument, it follows that any two such paths obtained from parallel full rows are vertex disjoint. Thus for all $1 \leq i \leq d$, we can obtain a set of vertex disjoint paths Q_i in G that traverse the points in the full rows of P' parallel to the i^{th} axis.

Finally we define a subgraph H of G as follows: H consists of the points in X. Furthermore, for any pair of points $p, q \in X$ such that p and q differ only along one coordinate, say the i^{th} coordinate, and differ by exactly $c+1$, H also consists of the sub-path between p and q of the corresponding path in Q_i connecting p and q. Now contracting these paths in H between adjacent points in X results in a d-dimensional grid with $\frac{n}{\lceil (c+1) \rceil^d}$ points. Thus, we conclude that $\mathsf{tw}(G) = \Omega\big(\frac{n^{1-1/d}}{(c+1)^{d-1}}\big) = \Omega\big(\frac{n^{1-1/d}}{c^{d-1}}\big)$. ◀

2.2 A first attempt: The Sierpiński carpet

Consider a set of points X obtained by the following method: start with a $3^k \times 3^k$ integer grid for some $k \in \mathbb{N}$ and partition it into 9 subgrids of equal size. We delete all the points in the central subgrid and recurse on the remaining 8 subgrids. The recursion stops when we arrive at a subgrid containing a single point. This is a natural discrete variant of the Sierpiński carpet.

▶ **Theorem 14.** *Let δ be the fractal dimension of the set of points X obtained above. Then we have $\delta \geq \log_3 8$.*

Proof. We start by recalling the definition of fractal dimension of a set of points P. It is the infimum $\delta > 0$ such that for any $\varepsilon > 0$, for any ball B of radius $r \geq 2\varepsilon$ and for any ε-net N, we have $|B \cap N| = O((r/\varepsilon)^\delta)$.

As seen in the construction, the width of the grid reduces by $\frac{1}{3}$ at every step of recursion. Thus we may assume that the width is $\frac{1}{3^i}$ when we stop. Let $\varepsilon = \frac{1}{2} \cdot \frac{1}{3^i}$. Let $N = X$. We have $n = |X| = 8^i$ since every step of recursion is done on the remaining 8 subgrids. Let $r = \sqrt{2}$. Let x be the point on the top left corner of the grid. Then $|B(x, r) \cap N| = n = 8^i \leq m \cdot \left(\frac{r}{\varepsilon}\right)^\delta$ for some constant $m > 0$. This gives $8^i \leq m \cdot (\sqrt{2} \cdot 2 \cdot 3^i)^\delta$. Taking log on both sides, we get $\log 8 \leq \delta \log 3$. Thus, $\delta \geq \log_3 8$. ◀

▶ **Theorem 15.** *There exists a set of n points P with fractal dimension $\delta \in (1, 2)$, and some c-spanner G of P, where $c < 1 + \sqrt{2}$, such that $\mathsf{tw}(G) = \Theta(n^{1-\frac{1}{\delta}-\varepsilon})$, for some $\varepsilon > 0$.*

2.3 One treewidth-extremal fractal: A discretized Cantor crossbar

In this section we describe the construction of a pointset in $\mathbb{R}^2$ with a specific fractal dimension of $\frac{\log 6}{\log 3}$. This pointset is a discretized version of the *Cantor crossbar*. We generalize this construction in the next section to construct pointsets with arbitrary fractal dimension.

▶ **Theorem 16.** *Let $\delta = \frac{\log(6)}{\log(3)}$. Then for all $n_0 \in \mathbb{N}$, there exists a set $P \subset \mathbb{R}^2$ of $n \geq n_0$ points of fractal dimension at most δ, such that for any $c \geq 1$, any c-spanner G of P has $\mathsf{tw}(G) = \Omega\left(\frac{n^{1-\frac{1}{\delta}}}{c}\right)$.*

Construction of the discrete Cantor crossbar. To prove the above theorem, we consider the set of points obtained as follows. First we define $f(0)$ and $h(0)$ and $g(0)$ to be a single point. Then we inductively define $h(i)$,$g(i)$ and $f(i)$ as follows. To get $h(i)$ we start with a $3^i \times 3^i$ integer grid and subdivide it into nine $3^{i-1} \times 3^{i-1}$ integer grids. Then we remove all 3 sub-grids in the middle row and replace each of the 6 remaining sub-grids with copies of $h(i-1)$. To get $g(i)$ we again start with a $3^i \times 3^i$ integer grid and subdivide it into nine $3^{i-1} \times 3^{i-1}$ integer grids. Then we remove all 3 sub-grids in the middle column and replace each of the 6 remaining sub-grids with copies of $g(i-1)$. Finally to get $f(i)$ we start with a $3^i \times 3^i$ integer grid and subdivide it into nine $3^{i-1} \times 3^{i-1}$ integer grids. Then we remove the central sub-grid. We then replace the four corner sub-grids with copies of $f(i-1)$. We replace the middle sub-grid in the first and last rows with copies of $h(i-1)$ and we replace the middle sub-grids of the first and last columns with copies of $g(i-1)$ as depicted in Figure 6. The pointset we require is given by $f(k)$ where k is any positive integer. We have the following two lemmas regarding the pointset $f(k)$.

▶ **Lemma 17.** $|f(k)| \leq 2 \cdot 6^k$.

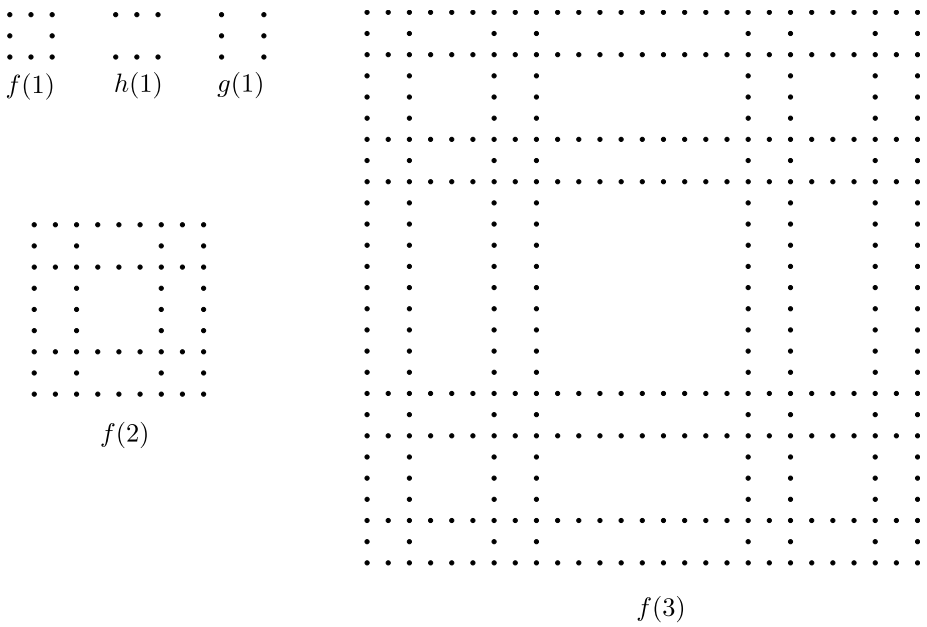


Figure 6 The construction of the discrete Cantor crossbar.

▶ **Lemma 18.** *Let $P = f(k)$. Then,* $\dim_{\mathsf{f}}(P) \leq \frac{\log(6)}{\log(3)}$.

▶ **Lemma 19.** *Let G be a c-spanner of P, where $c \geq 1$. Then,* $\mathsf{tw}(G) = \Omega\left(\frac{2^k}{c}\right)$.

Proof of Theorem 16. Using Lemma 18 and Lemma 19, we get that

$$\mathsf{tw}(G) = \Omega\left(\frac{(3^{k\delta})^{1-\frac{1}{\delta}}}{c}\right) = \Omega\left(\frac{n^{1-\frac{1}{\delta}}}{c}\right)$$

◀

.

2.4 A family of treewidth-extremal fractals for all dimensions

We can now generalize the ideas from the previous section to obtain a family of pointsets that allow us to a get a lower bound on the treewidth of spanners for any given choice of fractal dimension.

Construction of treewidth-extremal fractal pointsets. Consider the family of pointsets defined as follows: For all integers $d > 0$, we name the dimensions from $\{1, 2, \ldots, d\}$ in an arbitrary manner. For all odd integers l and v such that $l > v$, we define each of $f^{l,v,d}(0), h_1^{l,v,d}(0), \ldots, h_d^{l,v,d}(0)$ to be a single point. We inductively define $f^{l,v,d}(i), h_1^{l,v,d}(i), \ldots$ $\ldots, h_d^{l,v,d}(i)$ as follows: For $h_1^{l,v,d}(i)$, we start with a d-dimensional l^i integer grid and subdivide it to get l^d identical l^{i-1} d-dimensional integer subgrids. Now, along every dimension $j \in \{2, \ldots, d\}$, we remove all $l^{d-1}v$ subgrids in the middle v rows of the subgrids. We then replace each of the remaining $l(l-v)^{d-1}$ subgrids with copies of $h_1^{l,v,d}(i-1)$. The pointset obtained is $h_1^{l,v,d}(i)$. In general for any $m \in [d]$, we construct $h_m^{l,v,d}(i)$ as follows: we start with a d-dimensional l^i integer grid and subdivide it to get l^d identical l^{i-1} d-dimensional integer subgrids. Then, along every dimension $j \neq m$, we remove all $l^{d-1}v$ subgrids in the middle v rows of the subgrids. We replace each of the remaining $l(l-v)^{d-1}$ subgrids

with copies of $h_m^{l,v,d}(i-1)$. The pointset thus obtained is $h_m^{l,v,d}(i)$. In order to construct $f^{l,v,d}(i)$, we start with a d-dimensional l^i integer grid and subdivide it into l^d identical l^{i-1} d-dimensional integer subgrids. Let S denote the set of the central v^d subgrids. Note that S is a d-dimensional grid with side length $v(l^{i-1})$. Now along every dimension $m \in [d]$, there are $\frac{(l-v)v^{d-1}}{2}$ subgrids on each side of S. We remove S as well as all such $\frac{(l-v)v^{d-1}}{2}$ subgrids lying on either side of S along every dimension. We replace each of the $\left(\frac{l-v}{2}\right)^d$ sub-grids in the 2^d corners with copies of $f^{l,v,d}(i-1)$. Then along every dimension m, we replace each of the remaining $\left(\frac{l-v}{2}\right)^{d-1} v2^{d-1}$ subgrids with copies of $h_m^{l,v,d}(i-1)$. The pointset thus obtained is $f(i)$. To generate the pointset P mentioned in the statement of Theorem 1 we pick l and v such that $\left|\delta - \frac{\log(l(l-v)^{d-1})}{\log l}\right| \leq \varepsilon$. Such a pair of odd numbers always exists. Let $\delta' = \frac{(\log l(l-v)^{d-1})}{\log l}$. Then, we set P to be $f^{l,v,d}(k)$, where k is any positive integer. We have the following lemmas regarding pointset P.

▶ **Lemma 20.** *For all odd integers l and v such that $l > v$, and for all positive integers $k > 0$, we have that* $\dim_{\mathrm{f}}(f^{l,v,d}(k)) \leq \frac{(\log l(l-v)^{d-1})}{\log l}$.

▶ **Lemma 21.** $\dim_f(P) \leq \delta'$.

Proof. This follows from Lemma 20 when applied to our choice of l,v and P. ◀

▶ **Lemma 22.** *Let G be a c-spanner of P, where $c \geq 1$. Then,* $\mathsf{tw}(G) = \Omega\left(\frac{(l-v)^{k(d-1)}}{c^{d-1}}\right)$.

Proof of Theorem 1. Using Lemma 22 and Lemma 21, we get that

$$\mathsf{tw}(G) = \Omega\left(\frac{(l-v)^{k(d-1)}}{c^{d-1}}\right) = \Omega\left(\frac{l^{k\delta'\left(1-\frac{1}{\delta'}\right)}}{c^{d-1}}\right) = \Omega\left(\frac{n^{1-\frac{1}{\delta'}}}{c^{d-1}}\right).$$

This combined with the result of Lemma 21 proves the statement of the theorem. ◀

3 Running time lower bound for Independent Set of Unit Balls

In this section, we present the proof of Theorem 2. Our argument uses a reduction from a type of Constraint Satisfaction Problem called the Geometric Constraint Satisfaction Problem. The definitions in this section are taken from [19].

▶ **Definition 23** (The Constraint Satisfaction Problem)**.** [19] The input instance I of a constraint satisfaction problem is a triple (V, D, C), where V is a set of variables that can take values in the domain D, and C is a set of constraints, with each constraint being a pair $\langle s_i, R_i\rangle$ such that:

- s_i is a tuple of variables of size m_i.
- R_i is an m_i-ary relation over D.

A valid solution to the problem is an assignment of values from D to each of the variables in V such that for all constraints $\langle s_i, R_i\rangle$, the assignment for each tuple s_i is in R_i.

For our purposes, we only need to consider the case where the constraints are binary, or in other words, for all i, we have that $m_i = 2$. We may assume that the input size $|I|$ of a binary CSP instance is a polynomial in $|V|$ and $|D|$.

For an instance I of the Constraint Satisfaction Problem, the *primal graph* is a graph G with vertex set V and an edge between $u, w \in V$ if and only if there exists a constraint $\langle s_i, R_i\rangle \in C$, such that $s_i = (u, w)$.

Let $R[n, d]$ denote the d-dimensional grid with vertex set $[n]^d$ and let $\mathcal{R}_d$ denote the set of graphs $R[n, d]$ for all $n \geq 1$.

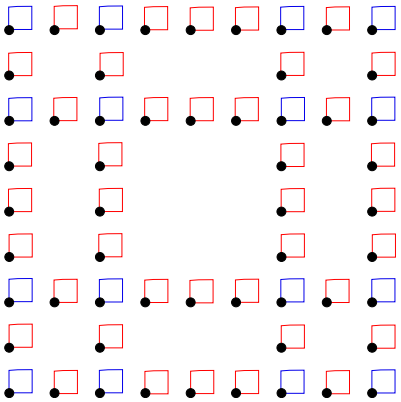


Figure 7 The case $d = 2$, $l = 3$, $v = 1$ and $m = 2$. Boxes contain centers of balls in B. The blue boxes are obtained from variables in I and the red boxes are obtained from variables in $I' \setminus I$.

▶ **Definition 24** (The $\leq$-CSP)**.** [19] A d-dimensional geometric $\leq$-CSP is a constraint satisfaction problem of the following form: The set of variables V is a subset of vertices of $R[n, d]$ for some n and the primal graph is an induced subgraph of $R[n, d]$. The domain is $[\Delta]^d$ for some integer $\Delta \geq 1$. The instance can contain arbitrary unary constraints but the binary constraints are of a special form. A geometric constraint is a constraint $\langle (\mathbf{a}, \mathbf{a}'), R \rangle$ with $\mathbf{a}' = \mathbf{a} + \mathbf{e}_i$ such that

$$R = \{((x_1, \ldots, x_d), (y_1, \ldots, y_d)) \mid x_i \leq y_i\}$$

This means that, if variables $\mathbf{a}$ and $\mathbf{a}'$ are adjacent with $\mathbf{a}'$ being larger by one in the i-th coordinate, then the i-th coordinate of the value of $\mathbf{a}$ is at most as large as the i-th coordinate of the value of $\mathbf{a}'$.

We will use the following theorem from [19] in the proof of Theorem 2. We remark that the condition $|V| = \Theta(n^d)$ is implicit in [19].

▶ **Theorem 25** ([19, Theorem 2.20])**.** *If for some fixed $d \geq 1$, there is an $f(|V|)n^{o(|V|^{1-1/d})}$ time algorithm for d-dimensional geometric $\leq$-CSP for some function f, where $|V| = \Theta(n^d)$, then ETH fails.*

Construction of $\leq$-CSP I'. Given $\delta \in (1, d)$ and $\varepsilon' > 0$, we can find odd integers l and v such that $\left|\delta - \frac{\log l(l-v)^{d-1}}{\log l}\right| \leq \varepsilon'$. Let $\delta' = \frac{\log l(l-v)^{d-1}}{\log l}$. Let I be a d-dimensional $\leq$-CSP instance with variables V and domain $[\Delta]^d$, where Δ is any positive integer. Let the primal graph of I be $R[n, d]$. We now define a new d-dimensional $\leq$-CSP I', with variables V' and domain $[\Delta]^d$, such that $|V'| = O\left(|V|^{\frac{d-1}{d} \cdot \frac{\delta'}{\delta'-1}}\right)$, and the primal graph of I' is $R[n_{new}, d]$, where $n_{new} = n^{O(1)}$. Let $m > 0$ be the smallest integer such that $l^{\frac{(m-1)(\delta'-1)}{d-1}} < n \leq l^{\frac{m(\delta'-1)}{d-1}}$. We construct $f^{l,v,d}(m)$, as in the proof of Theorem 1. From Lemma 22, we know that $f^{l,v,d}(m)$ contains a subset of points that form a d-dimensional grid of side length $l^{\frac{m(\delta'-1)}{d-1}}$. Let M denote this grid contained in $f^{l,v,d}(m)$. Since the variables V lie on the grid $R[n, d]$ and $n^d \leq l^{\frac{md(\delta'-1)}{d-1}}$, we can place the variables V on grid M such that their position relative to each other in M is the same as in $R[n, d]$. By abuse of notation, we refer to the subset of M containing variables V as V. We refer to the pointset $f^{l,v,d}(m) \setminus M$ as C. We observe

that the points in C connect adjacent points of M. We now define a new set of variables $V' = V \cup C$. If $\mathbf{a} \in V$, then the set of unary constraints for $\mathbf{a}$ is $R_{\mathbf{a}}$. If $\mathbf{a} \in C$, then $\mathbf{a}$ belongs to a chain of points that connects two points of M along some dimension i, where $i \in [d]$. For an $\mathbf{a} \in C$ connecting two points of M along dimension i, we define $R_{\mathbf{a}}$ as follows:

$$R_{\mathbf{a}} = \{(a_1, \ldots, a_{i-1}, a_i, a_{i+1} \ldots, a_d) \mid a_j = 0 \; \forall \; j \neq i \text{ and } a_i \in [\Delta]\}.$$

Thus for every $\mathbf{a} \in C$, we have $|R_{\mathbf{a}}| = \Delta$. We define $R_{\mathbf{a}}$ for all variables $\mathbf{a} \in C$ in a similar manner. The binary constraints on variables V' are described as follows: a binary constraint is a constraint $\langle(\mathbf{a}, \mathbf{a}'), R\rangle$, with $\mathbf{a}' = \mathbf{a} + \mathbf{e}_i$ in $f^{l,v,d}(m)$ such that

$$R = \{((x_1, \ldots, x_d), (y_1, \ldots, y_d)) \mid x_i \leq y_i\}.$$

Let I' denote the new $\leq$-geometric CSP with variables V', and unary and binary constraints as defined above. From the choice of m, we have that $|V'| \leq l^{\delta'} \cdot n^{(d-1)\cdot\frac{\delta'}{\delta'-1}}$. Thus $|V'| = O\left(|V|^{\frac{d-1}{d}\cdot\frac{\delta'}{\delta'-1}}\right)$, where we use the fact that δ, δ', and l are fixed constants. Similarly, we get that $n_{new} = n^{O(1)}$.

▶ **Lemma 26.** *I' is satisfiable if and only if I is satisfiable.*

Proof of Theorem 2. This proof is similar to the proof of Theorem 3.1 in [19]. Let I be a d-dimensional $\leq$-CSP with variables V and domain $[\Delta]^d$, and with primal graph $R[n, d]$. As explained before, we construct a new $\leq$-CSP I' with variables V' and domain $[\Delta]^d$, such that $|V'| = O\left(|V|^{\frac{d-1}{d}\cdot\frac{\delta}{\delta-1}}\right)$ and, the primal graph of I' is $R[n_{new}, d]$, where $n_{new} = n^{O(1)}$. We construct a set B of n_{new} balls, of diameter 1 each, such that the centers of the balls have fractal dimension at most δ', and B contains a set of $|V'|$ pairwise non-intersecting balls if and only if I' has a satisfying assignment.

If there is an $f(|V'|)n_{new}^{o(|V'|^{1-1/\delta'})}$ time algorithm for finding $|V'|$ pairwise non-intersecting balls in B, then we can solve I' in $f(|V'|)n_{new}^{o(|V'|^{1-1/\delta'})}$ time. Since I' is satisfiable if and only if I is satisfiable, we obtain a solution for I in time $f(|V'|)n_{new}^{o(|V'|^{1-1/\delta'})}$. Let h be a computable function such that $h(|V|) = f(|V'|)$. Since $f(|V'|)n_{new}^{o(|V'|^{1-1/\delta'})} = h(|V|)n^{o(|V|^{1-1/d})}$, we obtain a solution for I in time $h(|V|)n^{o(|V|^{1-1/d})}$. By Theorem 25, this contradicts ETH. Please refer to the full version of the paper for a detailed proof. ◀

▶ Remark 27. We can similarly show that assuming the Exponential Time Hypothesis, for any $\delta \in (1, d)$ and any $\varepsilon > 0$, the problem of finding k-pairwise non-intersecting d-dimensional axis parallel unit cubes in a collection of n cubes with centers having fractal dimension at most δ cannot be solved in time $f(k)n^{O(k^{1-1/(\delta-\varepsilon)})}$, for any computable function f. Given a $\leq$-CSP instance I, we reduce I to another $\leq$-CSP instance I' as explained in the beginning of this section. We then use the construction and analysis as in the proof of Theorem 3.2 of [19], replacing I with I'. Using Theorem 25 and the proof of Theorem 2, we get the desired result.

References

1 Jochen Alber and Jiří Fiala. Geometric separation and exact solutions for the parameterized independent set problem on disk graphs. In *Foundations of Information Technology in the Era of Network and Mobile Computing*, pages 26–37. Springer, 2002.

2 Yair Bartal, Lee-Ad Gottlieb, and Robert Krauthgamer. The traveling salesman problem: low-dimensionality implies a polynomial time approximation scheme. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 663–672. ACM, 2012.

3 Hubert TH Chan, Anupam Gupta, Bruce M Maggs, and Shuheng Zhou. On hierarchical routing in doubling metrics. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 762–771. Society for Industrial and Applied Mathematics, 2005.

4 T-H Hubert Chan and Anupam Gupta. Small hop-diameter sparse spanners for doubling metrics. *Discrete & Computational Geometry*, 41(1):28–44, 2009.

5 Chandra Chekuri and Julia Chuzhoy. Polynomial bounds for the grid-minor theorem. *Journal of the ACM (JACM)*, 63(5):40, 2016.

6 Richard Cole and Lee-Ad Gottlieb. Searching dynamic point sets in spaces with bounded doubling dimension. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 574–583. ACM, 2006.

7 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

8 Lee-Ad Gottlieb and Liam Roditty. An optimal dynamic spanner for doubling metric spaces. In *European Symposium on Algorithms*, pages 478–489. Springer, 2008.

9 Anupam Gupta and Kevin Lewi. The online metric matching problem for doubling metrics. In *International Colloquium on Automata, Languages, and Programming*, pages 424–435. Springer, 2012.

10 Sariel Har-Peled and Manor Mendel. Fast construction of nets in low-dimensional metrics and their applications. *SIAM Journal on Computing*, 35(5):1148–1184, 2006.

11 T-H Hubert Chan and Anupam Gupta. Approximating tsp on metrics with bounded global growth. *SIAM Journal on Computing*, 41(3):587–617, 2012.

12 David R Karger and Matthias Ruhl. Finding nearest neighbors in growth-restricted metrics. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 741–750. ACM, 2002.

13 Kyohei Kozawa, Yota Otachi, and Koichi Yamazaki. The carving-width of generalized hypercubes. *Discrete Math.*, 310(21):2867–2876, 2010. doi:10.1016/j.disc.2010.06.039.

14 Robert Krauthgamer and James R Lee. The black-box complexity of nearest-neighbor search. *Theoretical Computer Science*, 348(2):262–276, 2005.

15 Robert Krauthgamer and James R Lee. Algorithms on negatively curved spaces. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 119–132. IEEE, 2006.

16 Robert Krauthgamer, James R Lee, Manor Mendel, and Assaf Naor. Measured descent: A new embedding method for finite metrics. *Geometric & Functional Analysis GAFA*, 15(4):839–858, 2005.

17 Dániel Marx. Efficient approximation schemes for geometric problems. In *European Symposium on Algorithms*, pages 448–459. Springer, 2005.

18 Dániel Marx. Can you beat treewidth? *Theory OF Computing*, 6:85–112, 2010.

19 Dániel Marx and Anastasios Sidiropoulos. The limited blessing of low dimensionality: When 1-1/d is the best possible exponent for d-dimensional geometric problems. In *Proceedings of the Thirtieth Annual Symposium on Computational Geometry*, SOCG'14, pages 67:67–67:76, New York, NY, USA, 2014. ACM. doi:10.1145/2582112.2582124.

20 Neil Robertson, Paul Seymour, and Robin Thomas. Quickly excluding a planar graph. *Journal of Combinatorial Theory, Series B*, 62(2):323–348, 1994.

21 Neil Robertson and Paul D Seymour. Graph minors. v. excluding a planar graph. *Journal of Combinatorial Theory, Series B*, 41(1):92–114, 1986.

22 Jeffrey S Salowe. Construction of multidimensional spanner graphs, with applications to minimum spanning trees. In *Proceedings of the seventh annual symposium on Computational geometry*, pages 256–261. ACM, 1991.

23 Anastasios Sidiropoulos and Vijay Sridhar. Algorithmic interpretations of fractal dimension. In *33rd International Symposium on Computational Geometry, SoCG 2017, July 4-7, 2017, Brisbane, Australia*, volume 77 of *LIPIcs*, pages 58:1–58:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. URL: `http://www.dagstuhl.de/dagpub/978-3-95977-038-5`, `doi:10.4230/LIPIcs.SoCG.2017.58`.

24 Warren D Smith and Nicholas C Wormald. Geometric separator theorems and applications. In *Foundations of Computer Science, 1998. Proceedings. 39th Annual Symposium on*, pages 232–243. IEEE, 1998.

25 Hideki Takayasu. *Fractals in the physical sciences*. Manchester University Press, 1990.

26 Kunal Talwar. Bypassing the embedding: algorithms for low dimensional metrics. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 281–290. ACM, 2004.

27 Pravin M Vaidya. A sparse graph almost as good as the complete graph on points ink dimensions. *Discrete & Computational Geometry*, 6(3):369–381, 1991.

The Trisection Genus of Standard Simply Connected PL 4–Manifolds

Jonathan Spreer[1]
Institut für Mathematik, Freie Universität Berlin
Arnimallee 2, 14195 Berlin, Germany
jonathan.spreer@fu-berlin.de
https://orcid.org/0000-0001-6865-9483

Stephan Tillmann[2]
School of Mathematics and Statistics F07, The University of Sydney, NSW 2006 Australia
stephan.tillmann@sydney.edu.au
https://orcid.org/0000-0001-6731-0327

Abstract

Gay and Kirby recently introduced the concept of a trisection for arbitrary smooth, oriented closed 4–manifolds, and with it a new topological invariant, called the trisection genus. In this note we show that the $K3$ surface has trisection genus 22. This implies that the trisection genus of all standard simply connected PL 4–manifolds is known. We show that the trisection genus of each of these manifolds is realised by a trisection that is supported by a singular triangulation. Moreover, we explicitly give the building blocks to construct these triangulations.

2012 ACM Subject Classification Mathematics of computing → Combinatoric problems, Mathematics of computing → Geometric topology, Mathematics of computing → Graph coloring, Theory of computation → Computational geometry

Keywords and phrases combinatorial topology, triangulated manifolds, simply connected 4-manifolds, K3 surface, trisections of 4-manifolds

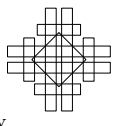
Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.71

Acknowledgements We would like to thank the anonymous reviewers for useful suggestions which improved the presentation of the paper.

1 Introduction

Gay and Kirby's construction of a *trisection* for arbitrary smooth, oriented closed 4–manifolds [13] defines a decomposition of the 4–manifold into three 4-dimensional handlebodies[3] glued along their boundaries in the following way: Each handlebody is a boundary connected sum of copies of $S^1 \times B^3$, and has boundary a connected sum of copies of $S^1 \times S^2$ (here, B^i denotes the i-dimensional ball and S^j denotes the j-dimensional sphere). The triple intersection of the 4-dimensional handlebodies is a closed orientable surface Σ, called the

[1] Research of the first author was supported by the Einstein Foundation (project "Einstein Visiting Fellow Santos").
[2] Research of the second author was supported in part under the Australian Research Council's Discovery funding scheme (project number DP160104502). The second author thanks the DFG Collaborative Center SFB/TRR 109 at TU Berlin, where parts of this work have been carried out, for its hospitality.
[3] A d-dimensional handlebody (or, more precisely, 1–handlebody) is the regular neighbourhood of a graph embedded into Euclidean d-space.




central surface, which divides each of their boundaries into two 3–dimensional handlebodies (and hence is a Heegaard surface). These 3–dimensional handlebodies are precisely the intersections of pairs of the 4–dimensional handlebodies.

A trisection naturally gives rise to a quadruple of non-negative integers $(g; g_0, g_1, g_2)$, encoding the genus g of the central surface Σ and the genera g_0, g_1, and g_2 of the three 4–dimensional handlebodies. The *trisection genus* of M, denoted $g(M)$, is the minimal genus of a central surface in any trisection of M. A trisection with $g(\Sigma) = g(M)$ is called a *minimal genus trisection*.

For example, the standard 4–sphere has trisection genus equal to zero, the complex projective plane equal to one, and $S^2 \times S^2$ equal to two. Meier and Zupan [19] showed that there are only six 4–manifolds of trisection genus at most two. An extended set of examples of trisections of 4–manifolds can be found in [13]. The recent works of Gay [14], and Meier, Schirmer and Zupan [16, 19, 18] give some applications and constructions arising from trisections of 4–manifolds and relate them to other structures on 4–manifolds.

A key feature is the so-called *trisection diagram* of the 4–manifold, comprising three sets of simple closed curves on the 2–dimensional central surface from which the 4–manifold can be reconstructed and various invariants of the 4–manifold can be computed. This is particularly interesting in the case where the central surface is of minimal genus, giving a minimal representation of the 4–manifold.

An approach to trisections using triangulations is given by Rubinstein and the second author in [23], and used by Bell, Hass, Rubinstein and the second author in [2] to give an algorithm to compute trisection diagrams of 4–manifolds, using a modified version of the input triangulation. The approach uses certain partitions of the vertex set (called *tricolourings*), and the resulting trisections are said to be *supported* by the triangulation. We combine this framework with a greedy-type algorithm for collapsibility to explicitly calculate trisections from existing triangulations of 4–manifolds. In doing so we are able to prove the following statement about the *K3 surface*, a 4–manifold that can be described as a quartic in $\mathbb{C}P^3$ given by the equation

$$z_0^4 + z_1^4 + z_2^4 + z_3^4 = 0.$$

▶ **Theorem 1.** *The trisection genus of the $K3$ surface is 22, that is, it is equal to its second Betti number.*

The algorithms of [2] can in particular be applied to a minimal trisection of the $K3$ surface to compute a minimal trisection diagram. This straightforward application is not done in this paper. Instead we focus on the following application of our result:

We already know that the trisection genera of $\mathbb{C}P^2$ and $S^2 \times S^2$ are equal to their respective second Betti numbers. Moreover, the second Betti number is additive, and the trisection genus is subadditive under taking connected sums. It thus follows from Theorem 1 and Lemma 6 that the trisection genus of every 4–manifold which is a connected sum of arbitrarily many (PL standard) copies of $\mathbb{C}P^2$, $S^2 \times S^2$ and the $K3$ surface must be equal to its second Betti number. This family includes the standard 4–sphere as the empty connected sum.

We refer to each member of this family of manifolds as a *standard simply connected PL 4–manifold*. Due to work by Freedman [11], Milnor and Husemoller [20], Donaldson [8], Rohlin [22], and Furuta [12], each of these manifolds must be homeomorphic to one of

$$k(\mathbb{C}P^2)\#m(\overline{\mathbb{C}P^2}),\ k(K3)\#m(S^2 \times S^2),\ \text{or } k(\overline{K3})\#m(S^2 \times S^2), \quad k, l, m, r \geq 0,$$

where $\overline{X}$ denotes X with the opposite orientation. Furthermore, modulo the 11/8-conjecture, standard simply connected PL 4–manifolds comprise all topological types of PL simply connected 4–manifolds. See [24, Section 5] for a more detailed discussion of the above statements, and see Section 2.1 for more details about 4–manifolds.

By analysing a particular family of singular triangulations of $\mathbb{C}P^2$, $S^2 \times S^2$ and the $K3$ surface due to Basak and the first author [1], we are able to prove the following even stronger statement.

▶ **Theorem 2.** *Let M be a standard simply connected PL 4–manifold. Then M has a trisection of minimal genus supported by a singular triangulation of M.*

The singular triangulations used as building blocks (connected summands) for this construction are all highly regular and come from what is called *simple crystallisations* (see Section 2.5 for details). They are explicitly listed in Appendix A.

In addition, we run a variant of the algorithm from [2] on the 440 495 triangulations of the 6-pentachoron census of closed 4–manifolds [4]. Surprisingly, while only 445 of these triangulations admit at least one trisection supported by its simplicial structure, some triangulations from the census admit as many as 48 of them.

The paper is organised as follows. In Section 2 we briefly go over some basic concepts used in the article, some elementary facts on trisections, as well as how to construct trisections from triangulations of 4–manifolds. In Section 3 we give details on the algorithm to compute trisections from a given triangulation and we present data obtained from running the algorithm on the 6–pentachora census of closed 4–manifolds. In Section 4 we then prove Theorems 1 and 2.

2 Preliminaries

2.1 Manifolds and triangulations

We assume basic knowledge of geometric topology and in particular manifolds. For a definition of homology groups and Betti numbers of a manifold see [15], for an introduction into simply connected 4–manifolds and their intersection forms, see [24].

In dimensions ≤ 4, there is a bijective correspondence between isotopy classes of smooth and piecewise linear structures [6, 7]. In this paper, all manifolds and maps are assumed to be piecewise linear (PL) unless stated otherwise. Our results apply to any compact smooth manifold by passing to its unique piecewise linear structure [26].

Recall that a manifold M is *simply connected* if every closed loop in M can be contracted to a single point (in other words, the fundamental group of M is trivial). While in dimensions 2 and 3 the only simply connected closed manifolds are the 2– and 3–sphere respectively, in dimension 4 there are infinitely many such manifolds.

Given two oriented simply connected 4–manifolds M and N, their *connected sum*, written $M\#N$, is defined as follows. First remove a 4-ball from both M and N and then glue the resulting manifolds along their boundaries via an orientation reversing homeomorphism. The resulting manifold is again oriented. The second Betti number is additive and the property of being simply connected is preserved under taking connected sums.

In this article, simply connected 4–manifolds are presented in the form of *singular triangulations*. A singular triangulation $\mathcal{T}$ of a (simply connected) 4–manifold M is a collection of $2n$ abstract pentachora together with $5n$ *gluing maps* identifying their $10n$ tetrahedral faces in pairs such that the underlying set of the quotient space $|\mathcal{T}|$ is PL-homeomorphic to M. The gluing maps generate an equivalence relation on the faces of the

pentachora, and an equivalence class of faces is referred to as a single face of the triangulation $\mathcal{T}$. Faces of dimension zero are called *vertices* and faces of dimension one are referred to as *edges* or the triangulation. The triangulation is *simplicial* if each equivalence class of faces has at most one member from each pentachoron; i.e. no two faces of a pentachoron are identified in $|\mathcal{T}|$. Every simplicial triangulation is also singular, and the second barycentric subdivision of a singular triangulation is always simplicial. The triangulation is *PL* if, in addition, the boundary of a small neighbourhood of every vertex is PL homeomorphic to a standard sphere. The number $2n$ of pentachora of $\mathcal{T}$ is referred to as the size of $\mathcal{T}$.

2.2 Trisections

We begin with a formal definition of a trisection of a 4–manifold.

▶ **Definition 3** (Trisection of closed manifold)**.** Let M be a closed, connected, piecewise linear 4–manifold. A *trisection* of M is a collection of three piecewise linear codimension zero submanifolds $H_0, H_1, H_2 \subset M$, subject to the following four conditions:

1. Each H_i is PL homeomorphic to a standard piecewise linear 4–dimensional 1–handlebody of genus g_i.
2. The handlebodies H_i have pairwise disjoint interior, and $M = \bigcup_i H_i$.
3. The intersection $H_i \cap H_j$ of any two of the handlebodies is a 3–dimensional 1–handlebody.
4. The common intersection $\Sigma = H_0 \cap H_1 \cap H_2$ of all three handlebodies is a closed, connected surface, the *central surface*.

The submanifolds $H_{ij} = H_i \cap H_j$ and Σ are referred to as the *trisection submanifolds*. In our illustrations, we use the colours blue, red, and green instead of the labels 0, 1, and 2 and we refer to $H_{\text{blue red}} = H_{br}$ as the *green* submanifold and so on.

The trisection in the above definition is also termed a $(g; g_0, g_1, g_2)$–trisection, where $g = g(\Sigma)$ is the genus of the central surface Σ. This definition is somewhat more general than the one originally given by Gay and Kirby [13] in that they ask for the trisection to be *balanced* in the sense that each handlebody H_i has the same genus. It is noted in [17, 23] that any unbalanced trisection can be stabilised to a balanced one.[4]

▶ **Definition 4** (Trisection genus)**.** The *trisection genus* of M, denoted $g(M)$, is the minimal genus of a central surface in any trisection of M. A trisection with $g(\Sigma) = g(M)$ is called a *minimal genus trisection*.

2.3 Three elementary facts

▶ **Lemma 5.** *If M has a trisection in which all 4–dimensional handlebodies are 4–balls, then the trisection is a minimal genus trisection.*

Proof. Suppose M has a $(g'; 0, 0, 0)$–trisection and a $(g; g_0, g_1, g_2)$–trisection. Moreover, suppose that $g = g(M)$. By [13], these have a common stabilisation. Suppose this is a $(g''; k_0, k_1, k_2)$–trisection. Each elementary stabilisation increases the genus of one handlebody and the genus of the central surface by one. Hence $g'' = g' + k_0 + k_1 + k_2$ and $g'' =$

[4] A stabilisation of a trisection with central surface of genus g is obtained by attaching a 1-handle to a 4-dimensional handlebody along a properly embedded boundary parallel arc in the 3-dimensional handlebody that is the intersection of the other two 4-dimensional handlebodies of the trisection. The result is a new trisection with central surface of genus g + 1, and exactly one of the 4-dimensional handlebodies has its genus increased by one.

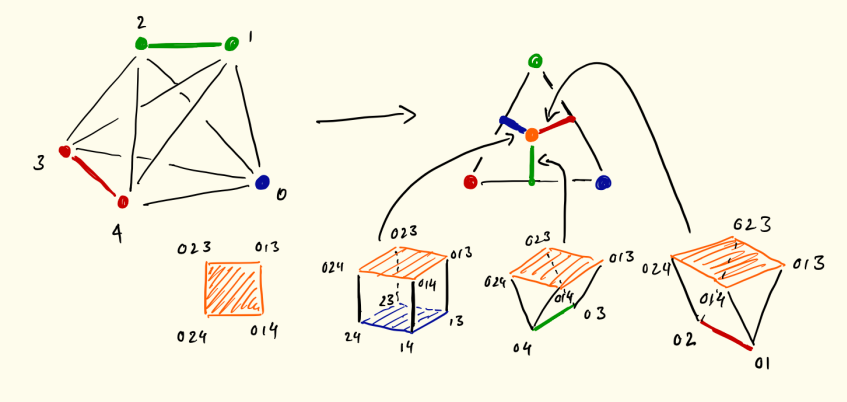


Figure 1 Pieces of the trisection submanifolds. The vertices of the pieces are barycenters of faces and labelled with the corresponding vertex labels. The central surface meets the pentachoron in a square. Two of the 3–dimensional trisection submanifolds meet the pentachoron in triangular prisms and the third (corresponding to the singleton) meets it in a cube. Moreover, any two of these meet in the square of the central surface.

$g + (k_0 - g_0) + (k_1 - g_1) + (k_2 - g_2)$. This gives $g' \geq g(M) = g = g' + g_0 + g_1 + g_2$. This forces $g_0 = g_1 = g_2 = 0$ and $g = g'$. ◀

▶ **Lemma 6.** *Suppose M has a $(g; g_0, g_1, g_2)$–trisection. Then $g \geq \dim H_2(M)$.*

Proof. In [9] it is shown how to compute the homology of a 4–manifold from a trisection diagram. Equation (3.9) in [9] in particular implies that $g \geq \dim H_2(M)$. ◀

▶ **Lemma 7.** *If M_0 has a trisection of genus g_0 and M_1 has a trisection of genus g_1, then the connected sum of M_0 and M_1 has a trisection of genus $g_0 + g_1$. In particular, $g(M_0 \# M_1) \leq g(M_0) + g(M_1)$.*

Proof. As explained in [13, §2], this follows by choosing, for the connect sum operation, two standardly trisected 4–balls in both 4–manifolds. ◀

2.4 Algorithms to compute trisections

Our set-up for algorithms to compute trisections follows [2], where a trisection on M is induced by *tricolourings* of the triangulation. This is now summarised; the reader is referred to [2] for a more detailed discussion.

Let M be a closed, connected 4–manifold with (not necessarily simplicial, but possibly singular) triangulation $\mathcal{T}$. A partition $\{P_0, P_1, P_2\}$ of the set of all vertices of $\mathcal{T}$ is a *tricolouring* if every 4–simplex meets two of the partition sets in two vertices and the remaining partition set in a single vertex. In this case, we also say that the triangulation is *tricoloured*.

Denote the vertices of the standard 2–simplex Δ^2 by v_0, v_1, and v_2. A tricolouring determines a natural map $\mu\colon M \to \Delta^2$ by sending the vertices in P_k to v_k and extending this map linearly over each simplex. Note that the pre-image of v_k is a graph Γ_k in the 1–skeleton of M spanned by the vertices in P_k.

The strategy in [2, 23] is to use μ to pull back the *cubical structure* of the simplex to a trisection of M. The *dual spine* Π^n in an n–simplex Δ^n is the $(n-1)$-dimensional subcomplex of the first barycentric subdivision of Δ^n spanned by those vertices of the first barycentric subdivision which are not vertices of Δ^n itself. This is shown for $n = 2$ and $n = 3$ in Figure 2.

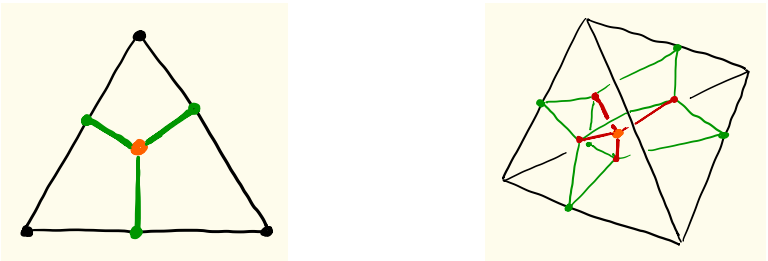


Figure 2 Dual cubes. Left: $\Pi^2 \subset \Delta^2$. Right: $\Pi^3 \subset \Delta^3$.

Decomposing along Π^n gives Δ^n a natural *cubical structure* with $n+1$ cubes of dimension n, and the lower-dimensional cubes that we focus on are the intersections of non-empty collections of these top-dimensional cubes.

Recall that a compact subpolyhedron P in the interior of a manifold M is called a *(PL) spine of* M if M collapses onto P. If P is a spine of M, then $M \setminus P$ is PL homeomorphic with $\partial M \times [0,1)$.

The pre-images under μ of the dual cubes of $\Pi^2 \subset \Delta^2$ have very simple combinatorics. The barycenter of Δ^2 pulls back to exactly one 2–cube in each pentachoron of M, and these glue together to form a surface Σ in M. This surface is the common boundary of each of the three 3–manifolds obtained as pre-images of an interior 1–cube (edge) of Δ^2. Each such 3–manifold is made up of cubes and triangular prisms, as in Figure 1. Each interior 1–cube c has boundary the union of the barycenter of Δ^2 and the barycenter b of an edge of Δ^2. Since the map $\mu\colon M \to \Delta^2$ is linear on each simplex, the pre-image $\mu^{-1}(c)$ collapses to the pre-image $\mu^{-1}(b)$. In particular, each 3–manifold has a spine made up of 1–cubes and 2–cubes.

It is shown in [2] that the above construction gives a trisection if:

1. the graph Γ_k is connected for each k; and
2. the pre-image of an interior 1–cube of Δ^2 has a 1–dimensional spine.

A tricolouring is a *c-tricolouring* if Γ_k is connected for each k. A c-tricolouring is a *ts-tricolouring* if the pre-image of each interior 1–cube collapses onto a 1–dimensional spine. In this case, the dual cubical structure of Δ^2 pulls back to a trisection of M.

▶ **Definition 8** (Trisection supported by triangulation)**.** We say that a trisection of M is *supported* by the triangulation $\mathcal{T}$ of M, if $\mathcal{T}$ is ts-tricolourable and the trisection is isotopic to the pull-back of the dual cubical structure of Δ^2. In this case, the trisection is said to be *dual* to the corresponding ts-tricolouring.

An algorithm to construct a ts-tricolouring from an arbitrary initial triangulation is given in [2]. It uses subdivisions and bistellar moves in order to justify that the resulting triangulation has a ts-tricolouring. As a result, the number of pentachora of the final triangulation is larger by a factor of 120 compared to the number of pentachora of the initial one (see [2, Theorem 4].)

The new algorithmic contribution of this paper is the development and implementation of an algorithm to determine whether a given triangulation directly admits a ts-tricolouring. This is given in §3.

2.5 Crystallisations of simply connected 4–manifolds

Let $G = (V, E)$ be a multigraph without loops. An *edge colouring* of G is a surjective map $\gamma\colon E \to C = \{0, 1, \dots, d\}$ such that $\gamma(e) \neq \gamma(f)$ whenever e and f are adjacent. The tuple (G, γ) is said to be a $(d+1)$-*coloured graph*. For the remainder of this section we fix d to be 4.

For a 5-coloured graph (G, γ), and $B \subseteq C$, $|B| = k$, the graph $G_B = (V(G), \gamma^{-1}(B))$ together with the colouring γ restricted to $\gamma^{-1}(B)$ is a k-coloured graph. If for all $j \in C$, $G_{C\setminus\{j\}}$ is connected, then (G, γ) is called *contracted*.

A 5-coloured graph (G, γ) defines a 4-dimensional simplicial cell complex $K(G)$: For each $v \in V(G)$ take a pentachoron Δ_v and label its vertices by $0, 1, 2, 3, 4$. If $u, v \in V(G)$ are joined by an edge e in G and $\gamma(e) = j$, we identify Δ_u and Δ_v along the tetrahedra opposite to vertex j with equally labelled vertices glued together. This way, no face of $K(G)$ can have self-identifications and $K(G)$ is a regular simplicial cell complex. We say that (G, γ) *represents* the simplicial cell complex $K(G)$. Since, in addition, the number of j-labelled vertices of $K(G)$ is equal to the number of components of $G_{C\setminus\{j\}}$ for each $j \in C$, the simplicial cell complex $K(G)$ contains exactly 5 vertices if and only if G is contracted [10].

For a 4–manifold M we call a 5-coloured contracted graph (G, γ) a *crystallisation* of M if the simplicial cell complex $K(G)$ is a singular triangulation of M (with no self-identifications of any of its faces). Every PL 4–manifold admits such a crystallisation due to work by Pezzana [21].

We refer to (G, γ) as simple, if $K(G)$ has exactly 10 edges. That is, if the one-skeleton of $K(G)$ is equal to the 1-skeleton of any of its pentachora. While not every simply connected PL 4–manifold can admit a simple crystallisation[5], this is true for the standard ones, see the article by Basak and the first author [1] for an explicit construction.

3 Implementation and data

In this section we describe the algorithm to check whether a singular triangulation admits a trisection and present some experimental data for the Budney-Burton census of singular triangulations of 4–manifolds up to six pentachora [4].

3.1 Implementation

Given a singular triangulation $\mathcal{T}$ with vertex set $V = V(\mathcal{T})$, perform the following three basic and preliminary checks.

1. Check if $\mathcal{T}$ has at least three vertices.
2. For all triangles $t \in \mathcal{T}$, check if t contains at least two vertices.
3. For all partitions of the vertex set into three non-empty parts $P_0 \sqcup P_1 \sqcup P_2 = V$, for all triangles $t \in \mathcal{T}$, check that no triangle is monochromatic with respect to (P_0, P_1, P_2) (i.e., every triangle contains vertices from at least two of the P_i, $i = 0, 1, 2$).

If any of these checks fail we conclude that $\mathcal{T}$ does not admit a tricolouring. Otherwise, for each tricolouring of $\mathcal{T}$ we proceed in the following way.

Compute the spines of the 4-dimensional handlebodies, that is, the graphs Γ_i in the 1-skeleton of $\mathcal{T}$ spanned by the vertices in P_i, $i = 0, 1, 2$. If all of the Γ_i, $i = 0, 1, 2$, are

[5] Note that, on the one hand there exist simply connected topological 4–manifolds with an infinite number of PL structures, and on the other hand, every topological type of simply connected PL 4–manifold can only admit a finite number of simple crystallisations.

connected, (P_0, P_1, P_2) defines a c-tricolouring of $\mathcal{T}$. Otherwise we conclude that the partition (P_0, P_1, P_2) is not a c-tricolouring of $\mathcal{T}$.

A spine of the 3-dimensional trisection submanifolds is given by the 1- and 2-cubes sitting in bi-coloured triangles and tetrahedra. Denote these three complexes γ_i, $i = 0, 1, 2$ (with γ_i being disjoint of all faces containing vertices in P_i). We compute the Hasse diagram of each of the γ_i and check their connectedness. In case all three complexes are connected, we perform a greedy-type collapse (as defined in [3]) to check whether γ_i collapses onto a 1-dimensional complex (and thus the 3-dimensional trisection submanifolds are handlebodies). Note that this is a deterministic procedure, see, for instance, [25].

If all of these checks are successful, (P_0, P_1, P_2) defines a ts-tricolouring and thus a trisection of M dual to this ts-tricolouring of $\mathcal{T}$. Otherwise we conclude that (P_0, P_1, P_2) does not define a ts-tricolouring.

Finally we compute the central surface, its genus, the genera of the handlebodies Γ_i and γ_i, $i = 0, 1, 2$, and, as a plausibility check, compare the genus of the central surface to the genera of the 3-dimensional handlebodies.

For $\mathcal{T}$ a v-vertex, n-pentachora triangulation the running time of this procedure is roughly the partition number $p(v)$ times a small polynomial in n. Since the triangulations we need to consider usually come with a very small number of vertices, the running time of our algorithm is sufficiently feasible for our purposes.

3.2 Experimental data

The census of singular triangulations of orientable, closed 4–manifolds [4] contains 8 triangulations with two pentachora, 784 triangulations with four pentachora and 440 495 triangulations with 6 pentachora.

In the 2-pentachora case, exactly 6 of the 8 triangulations have at least 3 vertices, and in exactly 3 of them all triangles contain at least two vertices. Overall, these three triangulations admit 19 tricolourings, 18 of which are c-tricolourings covering 2 triangulations. Of these 18 c-tricolourings all are dual to trisections. Overall, exactly 2, that is, 25% of all 2-pentachora triangulations admit a trisection.

Of the 784 triangulations with 4-pentachora, exactly 324 have at least three vertices, and exactly 24 only have triangles containing at least two vertices. Each of these 24 triangulations admits at least one tricolouring. Altogether they admit a total of 88 tricolourings. In 15 triangulations, at least one tricolouring defines connected graphs Γ_i, $0, 1, 2$, and is thus a c-tricolouring. These 15 triangulations admit a total of 72 c-tricolourings. All 72 of them are dual to trisections giving rise to a total of 15 triangulations (1.9% of the census) in the 4-pentachora census admitting a trisection.

There are 440 495 triangulations in the 6-pentachoron census, exactly 116 336 of which have at least three vertices, and exactly 837 of which have no triangles with only one vertex. 824 of these 837 triangulations admit at least one tricolouring, summing up to a total of 1 689 tricolourings in the census. Amongst these, 450 triangulations have at least one c-tricolouring, summing up to a total of 1 100 c-tricolourings. In 8 of these 1 100 c-tricolourings, the 2-complexes γ_i, $i = 0, 1, 2$, are not all connected, and in a further 5 of them not all γ_i, $i = 0, 1, 2$, collapse to a 1-dimensional complex. The remaining $1,087$ ts-tricolourings occur in 445 distinct triangulations. It follows that a total of 445 (or 0.1%) of all 6-pentachora triangulations admit a trisection.

Table 1 gives an overview of how many tricolourings, c-tricolourings, and ts-tricolourings can be expected from a triangulation admitting at least one tricolouring. As can be observed

Table 1 Number of triangulations in the 6-pentachora census having k tricolourings ($\mathfrak{n}_{\mathrm{tc}}$ in second column), c-tricolourings ($\mathfrak{n}_{\mathrm{c}}$ in third column), and ts-tricolourings ($\mathfrak{n}_{\mathrm{ts}}$ in fourth column).

k	$\mathfrak{n}_{\mathrm{tc}}$	$\mathfrak{n}_{\mathrm{c}}$	$\mathfrak{n}_{\mathrm{ts}}$
1	518	248	243
2	155	76	79
3	67	56	55
4	24	36	34
5	22	0	0
6	15	16	16
8	4	6	6
9	4	0	0
10	3	1	1
11	1	0	0
12	2	2	2
15	4	4	5
18	1	1	0
24	0	2	2
27	2	0	0
36	0	1	1
48	1	1	1
51	1	0	0
Σ	824	450	445

from the table, there exist triangulations in the census with as many as 51 distinct tricolourings, and 48 distinct ts-tricolourings.

For many of the triangulations with multiple ts-tricolourings, all trisections dual to them have central surfaces of the same genus and 4-dimensional handlebodies of the same genera. For some, however, a smallest genus trisection occurs amongst a number of trisections of higher genera. For instance, one of the largest ranges of genera of central surfaces is exhibited by a triangulation with 15 ts-tricolourings, supporting trisections of type $(0;0,0,0)$ $(\times 10)$, $(1;1,0,0)$ $(\times 4)$, and $(2;1,1,0)$ $(\times 1)$. The triangulation is necessarily homeomorphic to S^4. Its isomorphism signature is given in Appendix A.

4 Trisection genus for all known standard simply connected PL 4–manifolds

This section contains the proofs of Theorems 1 and 2. We start with a purely theoretical observation.

▶ **Lemma 9.** *Let M be a simply connected 4-manifold with second Betti number β_2, and let $\mathcal{T}$ be a triangulation of M coming from a simple crystallisation. Then $\mathcal{T}$ admits 15 c-tricolourings.*

Moreover, if some of these c-tricolourings are in fact ts-tricolourings, then their dual trisections must be of type $(\beta_2;0,0,0)$. In particular they must all be minimal genus trisections.

Proof. A triangulation of a 4–manifold M coming from a simple crystallisation is a simplicial cell complex $\mathcal{T}$ such that a) $M \cong_{PL} |\mathcal{T}|$, b) no face has self-identifications, and c) every

4-simplex shares the same 5 vertices and the same 10 edges.

Given such a triangulation $\mathcal{T}$, property c) ensures, that there are 15 ways to properly tricolour the 5 vertices of $\mathcal{T}$ (one colour colours a single vertex and the remaining two colours colour the remaining four vertices in pairs), and each one of them colours every 4-simplex equally and thus in a valid way. Moreover, also because of property c), every colour either spans a single edge or a single vertex of $\mathcal{T}$, and because of property b) a neighbourhood of this edge or vertex is diffeomeomorphic to a 4-ball. In particular, M decomposes into three 4-balls (i.e., handlebodies of genus zero) and all possible 15 tricolourings from above are in fact c-tricolourings.

Hence, let us assume that the 3-dimensional trisection submanifolds dual to some of the c-tricolourings are handlebodies – and thus some of the c-tricolourings are, in fact, ts-tricolourings. It remains to determine the genus of the central surface of the trisection dual to these ts-tricolourings.

The f-vector of $\mathcal{T}$ is given by

$$f(\mathcal{T}) = (5, 10, 10\beta_2 + 10, 15\beta_2 + 5, 6\beta_2 + 2),$$ [6]

where $\beta_2 = \dim(H_2(M, \mathbb{Z}))$. Every pair of the 5 vertices spans exactly one of the 10 edges, and each of the 10 boundaries of triangles spanned by the 10 edges of $\mathcal{T}$ bounds exactly $(\beta_2 + 1)$ parallel copies of triangles. It follows that there are exactly $4\beta_2 + 4$ tricoloured triangles. The central surface is thus spanned by $6\beta_2 + 2$ quadrilaterals, $4\beta_2 + 4$ vertices, and is (orientable) of Euler characteristic $2 - 2\beta_2$. Hence, its genus is equal to the second Betti number of M.

The trisection must be a minimal genus trisection for (at least) two reasons: (i) all of its 4-dimensional handlebodies are of genus zero (Lemma 5), and (ii) the genus of its central surface equals the second Betti number of M (Lemma 6). ◀

Proof of Theorem 1. In [1], Basak and the first author constructively show that there exists a simple crystallisation of the $K3$ surface. The result now follows from Lemma 9 together with a check of the collapsibility of the 3-dimensional trisection submanifolds for a particular example triangulation coming from a simple crystallisation.

The isomorphism signature given in Appendix A belongs to such a singular triangulation of the $K3$ surface supporting a trisection. ◀

Proof of Theorem 2. Given an arbitrary standard simply connected 4-manifold M and a simple crystallisation of M, Lemma 9 tells us that its associated triangulation $\mathcal{T}$ admits 15 c-tricolourings. It remains to show for a particular such triangulation of M and its 15 c-tricolourings that every 3-dimensional trisection submanifold is a handlebody, that is, it retracts to a 1-dimensional complex.

For this, w.l.o.g. assume that for every choice of c-tricolouring, the colour colouring only one vertex is blue b, and the colours colouring two vertices are red r and green g. By construction, the two 3-dimensional trisection submanifolds defined by b and r (b and g) retract to the (multi-)graph whose vertices are the mid-points of the two edges with endpoints coloured by b and r (b and g), and whose edges are normal arcs parallel to the

[6] Property c) in the definition of a simple crystallisation states that $\mathcal{T}$ must have 5 vertices and 10 edges. Since all ten edges of $\mathcal{T}$ are contained in a single pentachoron, $M \cong_{PL} |\mathcal{T}|$ must be simply connected and thus its Euler characteristic is given by $\chi(M) = 2 + \beta_2(M, \mathbb{Z})$. The other entries of the f-vector are then determined by the Dehn–Sommerville equations for 4-manifolds.

monochromatic edge in the β_2+1 triangles coloured rrb (ggb respectively). In particular, these two 3-dimensional trisection submanifolds retract to a 1-dimensional complex of genus β_2.

The third 3-dimensional trisection submanifold defined by r and g initially retracts to a 2-dimensional subcomplex Q_{rg} with 4 vertices, one for every rg-coloured edge, $4\beta_2 + 4$ edges, one for each rrg- and each rgg-coloured triangle, and $3\beta_2 + 1$ quadrilaterals, one for each $rrgg$-coloured tetrahedron. This 2-dimensional complex might or might not continue to collapse to a 1-dimensional complex depending on the combinatorial properties of $\mathcal{T}$. If Q_{rg} collapses, however, it must collapse to a 1-dimensional complex of genus β_2.

The unique simple crystallisation of $\mathbb{C}P^2$ as well as 266 of the 267 simple crystallisations of $S^2 \times S^2$ translate to triangulations where all of the 15 c-tricolourings (guaranteed by Lemma 9) are in fact ts-tricolourings. Moreover, there exist various simple triangulations of the $K3$ surface with this property. See [1] for pictures of representative simple crystallisations of $\mathbb{C}P^2$, $S^2 \times S^2$ and $K3$. These particular simple crystallisations turn out to have associated triangulations with 15 ts-tricolourings. See Appendix A for their isomorphism signatures.

Assume that for all connected sums N of these three prime simply connected 4–manifolds with arbitrary orientation and second Betti number $\leq k$, there always exists a triangulation coming from a simple crystallisation with all 15 c-tricolourings being also ts-tricolourings.

It remains to show that for two such manifolds N_1 and N_2, there exists a triangulation coming from a simple crystallisation of $N_1 \# N_2$ with this property. By the induction hypothesis we can assume that there exist such triangulations $\mathcal{T}_i$ of N_i, $i = 1, 2$, with all 15 c-tricolourings producing 2-complexes $Q_{rg}(\mathcal{T}_i)$, $i = 1, 2$, collapsing to a 1-dimensional complex.

Fix a ts-tricolouring on $\mathcal{T}_1$ and a collapsing sequence of the quadrilaterals of $Q_{rg}(\mathcal{T}_1)$. Remove one of the two 4-simplices containing the quadrilateral which is collapsed last. Similarly, fix a ts-tricolouring and collapsing sequence on $\mathcal{T}_2$ and remove one of the two 4-simplices containing the quadrilateral removed first. Glue together both triangulations along their boundaries such that the edges through which the last quadrilateral of $Q_{rg}(\mathcal{T}_1)$ and the first quadrilateral of $Q_{rg}(\mathcal{T}_2)$ are collapsed, are aligned. For this, colours r and g of $\mathcal{T}_2$ might need to be swapped (note that such a swap in colours does not change the ts-tricolouring class as such). The collapsing sequence can now be concatenated, yielding a collapsing sequence for $Q_{rg}(\mathcal{T}_1 \# \mathcal{T}_2)$.

Moreover, the property of a triangulation of coming from a simple crystallisation is preserved under taking connected sums of the type as described above [1]. Hence, repeating this process for all 15 tricolourings finishes the proof. ◀

References

1. Biplab Basak and Jonathan Spreer. Simple crystallizations of 4-manifolds. *Adv. in Geom.*, 16(1):111–130, 2016. arXiv:1407.0752 [math.GT]. doi:10.1515/advgeom-2015-0043.
2. M. Bell, J. Hass, J. Hyam Rubinstein, and S. Tillmann. Computing trisections of 4-manifolds. *ArXiv e-prints*, nov 2017. arXiv:1711.02763.
3. Bruno Benedetti and Frank H. Lutz. Random discrete Morse theory and a new library of triangulations. *Exp. Math.*, 23(1):66–94, 2014.
4. Benjamin A. Burton and Ryan Budney. A census of small triangulated 4-manifolds. in preparation, ≥2017.
5. Benjamin A. Burton, Ryan Budney, William Pettersson, et al. Regina: Software for low-dimensional topology. http://regina-normal.github.io/, 1999–2017.
6. Stewart S. Cairns. Introduction of a Riemannian geometry on a triangulable 4-manifold. *Ann. of Math. (2)*, 45:218–219, 1944. doi:10.2307/1969264.

7 Stewart S. Cairns. The manifold smoothing problem. *Bull. Amer. Math. Soc.*, 67:237–238, 1961. doi:10.1090/S0002-9904-1961-10588-0.

8 S. K. Donaldson. An application of gauge theory to four-dimensional topology. *J. Differential Geom.*, 18(2):279–315, 1983. URL: http://projecteuclid.org/getRecord?id=euclid.jdg/1214437665.

9 P. Feller, M. Klug, T. Schirmer, and D. Zemke. Calculating the homology and intersection form of a 4-manifold from a trisection diagram. *ArXiv e-prints*, 2017. arXiv:1711.04762.

10 M. Ferri, C. Gagliardi, and L. Grasselli. A graph-theoretical representation of PL-manifolds—a survey on crystallizations. *Aequationes Math.*, 31(2-3):121–141, 1986. doi:10.1007/BF02188181.

11 M. Freedman. The topology of four-dimensional manifolds. *J. Differential Geom.*, 17:357–453, 1982.

12 M. Furuta. Monopole equation and the $\frac{11}{8}$-conjecture. *Math. Res. Lett.*, 8(3):279–291, 2001. doi:10.4310/MRL.2001.v8.n3.a5.

13 David Gay and Robion Kirby. Trisecting 4-manifolds. *Geom. Topol.*, 20(6):3097–3132, 2016. doi:10.2140/gt.2016.20.3097.

14 David T. Gay. Trisections of Lefschetz pencils. *Algebr. Geom. Topol.*, 16(6):3523–3531, 2016. doi:10.2140/agt.2016.16.3523.

15 A. Hatcher. *Algebraic Topology*. Cambridge University Press, 2002. URL: http://books.google.de/books?id=BjKs86kosqgC.

16 Jeffrey Meier, Trent Schirmer, and Alexander Zupan. Classification of trisections and the generalized property R conjecture. *Proc. Amer. Math. Soc.*, 144(11):4983–4997, 2016. doi:10.1090/proc/13105.

17 Jeffrey Meier, Trent Schirmer, and Alexander Zupan. Classification of trisections and the generalized property R conjecture. *Proc. Amer. Math. Soc.*, 144(11):4983–4997, 2016. doi:10.1090/proc/13105.

18 Jeffrey Meier and Alexander Zupan. Bridge trisections of knotted surfaces in S^4. *Trans. Amer. Math. Soc.*, 369(10):7343–7386, 2017. doi:10.1090/tran/6934.

19 Jeffrey Meier and Alexander Zupan. Genus-two trisections are standard. *Geom. Topol.*, 21(3):1583–1630, 2017. doi:10.2140/gt.2017.21.1583.

20 John Milnor and Dale Husemoller. *Symmetric bilinear forms*, volume 73 of *Ergebnisse der Mathematik und ihrer Grenzgebiete*. Springer-Verlag, New York, 1973.

21 Mario Pezzana. Sulla struttura topologica delle varietà compatte. *Ati Sem. Mat. Fis. Univ. Modena*, 23(1):269–277 (1975), 1974.

22 V. A. Rohlin. New results in the theory of four-dimensional manifolds. *Doklady Akad. Nauk SSSR (N.S.)*, 84:221–224, 1952.

23 J. Hyam Rubinstein and S. Tillmann. Multisections of piecewise linear manifolds. *ArXiv e-prints*, 2016. arXiv:1602.03279.

24 Nikolai Saveliev. *Lectures on the topology of 3-manifolds*. De Gruyter Textbook. Walter de Gruyter & Co., Berlin, revised edition, 2012. An introduction to the Casson invariant.

25 Martin Tancer. Recognition of collapsible complexes is np-complete. *Discrete & Computational Geometry*, 55(1):21–38, Jan 2016. doi:10.1007/s00454-015-9747-1.

26 J. H. C. Whitehead. On C^1-complexes. *Ann. of Math. (2)*, 41:809–824, 1940. doi:10.2307/1968861.

A Isomorphism signatures of triangulations of S^4, $\mathbb{C}P^2$, $S^2 \times S^2$ and the $K3$ surface

Isomorphism signatures of the triangulations associated to three simple crystallisations of $\mathbb{C}P^2$, $S^2 \times S^2$ and the $K3$ surface, and a triangulation homeomorphic to S^4. Each of them admits 15 ts-tricolourings (see Section 4 for details, see [1] for pictures).

To construct the triangulations download *Regina* [5], and produce a new 4-manifold triangulation by selecting type of triangulation "From isomorphism signature" and pasting in one of the strings given below (please note that you need to remove newline characters before pasting in the isomorphism signature of the $K3$ surface).

Unique 8-pentachora simple crystallisation of $\mathbb{C}P^2$

```
iLvLQQQkbghhghhfffggfaaaaaaaaaaaaaaaaaaaaaaaaaa
```

14-pentachora simple crystallisation of $S^2 \times S^2$

```
oLvMPLQAPMQPkbfgghjihhiilkkmllmnnnnaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

134-pentachora simple crystallisation of the $K3$ surface

```
-cgcLvLALLvvwwzvzMvwLAwvwvvvwvwAAPvPPwzQQwMvPAQLQzQzPwLvwvPzvPwQLAPAPQMQwMQQQQAQ
zQQQQQAQPQQQQQQQQQwzvQQMMMMQQQQQQQQPkcahafafakaiasauaxapaBaDaGaKaDaFaFaQaWa3a6a
1aTa7aYaSaTa-aVa4a9aVaab9abbabPaPaZaibPagbfb5a3a5a1aZa0a2a0a-aVa-a8ahbQaubvbubz
bAbGbIbHbHbNbObQbQbKbLbRbHbvbLbwbMbTbTbububWbGbXbHbObRbDbYbNbYbKbwbObSbGbIbVbMb
MbZbZbGbVb1bWbxbTbPbWbXb0b3bPbPb3bCbCb3bYb0bJbybybJb2bIbvbIbvb3bzbxbzb5bWb1b5bz
bAbDbRb4bFbDbSb2bUbUb2bUbUb4bKbCbybybCb8bac-b+b+b+b9b9b8b8bbc6bdc6becacbc9b9bdc
6bec-b7bdc-b-bdcecccaccc7b+b8bbc6b7b7bfcfcfcfcaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

Triangulation homeomorphic to S^4 supporting trisections of multiple types

Triangulation homeomorphic to S^4 with 15 ts-tricolourings, supporting trisections of type $(0; 0, 0, 0)$ $(\times 10)$, $(1; 1, 0, 0)$ $(\times 4)$, and $(2; 1, 1, 0)$ $(\times 1)$. See Section 3.2 for details.

```
gLAAMQbbcddeffffaaaaaaaaaaaaaaaaaaaa
```

An $O(n \log n)$-Time Algorithm for the k-Center Problem in Trees

Haitao Wang
Department of Computer Science, Utah State University
Logan, UT 84322, USA
haitao.wang@usu.edu

Jingru Zhang
Department of Computer Science, Marshall University
Huntington, WV 25705, USA
jingru.zhang@marshall.edu

Abstract

We consider a classical k-center problem in trees. Let T be a tree of n vertices and every vertex has a nonnegative weight. The problem is to find k centers on the edges of T such that the maximum weighted distance from all vertices to their closest centers is minimized. Megiddo and Tamir (SIAM J. Comput., 1983) gave an algorithm that can solve the problem in $O(n \log^2 n)$ time by using Cole's parametric search. Since then it has been open for over three decades whether the problem can be solved in $O(n \log n)$ time. In this paper, we present an $O(n \log n)$ time algorithm for the problem and thus settle the open problem affirmatively.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms, Theory of computation → Computational geometry

Keywords and phrases k-center, trees, facility locations

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.72

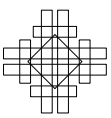
Related Version A full version of this paper is available at `https://arxiv.org/abs/1705.02752`

1 Introduction

In this paper, we study a classical k-center problem in trees. Let T be a tree of n vertices. Each edge $e(u, v)$ connecting two vertices u and v has a positive length $d(u, v)$, and we consider the edge as a line segment of length $d(u, v)$ so that we can talk about "points" on the edge. For any two points p and q of T, there is a unique path in T from p to q, denoted by $\pi(p, q)$, and by slightly abusing the notation, we use $d(p, q)$ to denote the length of $\pi(p, q)$. Each vertex v of T is associated with a weight $w(v) \geq 0$. The *k-center problem* is to compute a set Q of k points on T, called *centers*, such that the maximum weighted distance from all vertices of T to their closest centers is minimized, or formally, the value $\max_{v \in V(T)} \min_{q \in Q}\{w(v) \cdot d(v, q)\}$ is minimized, where $V(T)$ is the vertex set of T. Note that each center can be in the interior of an edge of T.

Kariv and Hakimi [20] first gave an $O(n^2 \log n)$ time algorithm for the problem. Jeger and Kariv [19] proposed an $O(kn \log n)$ time algorithm. Megiddo and Tamir [25] solved the problem in $O(n \log^2 n \log \log n)$ time, and the running time can be reduced to $O(n \log^2 n)$ by Cole's parametric search [12]. Some progress has been made very recently by Banik et al. [3] for small values of k, where an $O(n \log n + k \log^2 n \log(n/k))$-time algorithm and another $O(n \log n + k^2 \log^2(n/k))$-time algorithm were given.

Since Megiddo and Tamir's work [25], it has been open whether the problem can be solved in $O(n \log n)$ time. In this paper, we settle this three-decade long open problem

34th International Symposium on Computational Geometry (SoCG 2018).
Editors: Bettina Speckmann and Csaba D. Tóth; Article No. 72; pp. 72:1–72:15


Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

affirmatively by presenting an $O(n \log n)$-time algorithm. Note that the previous $O(n \log^2 n)$-time algorithm [12, 25] and the first algorithm in [3] both rely on Cole's parametric search, which involves a large constant in the time complexity due to the AKS sorting network [2]. Our algorithm, however, avoids Cole's parametric search.

If each center is required to be located a vertex of T, then we call it the *discrete* case. The previously best-known algorithm for this case runs in $O(n \log^2 n)$ time [26]. Our techniques also solve the discrete case in $O(n \log n)$ time.

Related work. Many variations of the k-center problem have been studied. If $k = 1$, then the problem is solvable in $O(n)$ time [23]. If T is a path, the k-center problem was already solved in $O(n \log n)$ time [9, 12, 25], and Bhattacharya and Shi [4] also gave an algorithm whose running time is linear in n but exponential in k.

For the unweighted case where the vertices of T have the same weight, an $O(n^2 \log n)$-time algorithm was given in [8] for the k-center problem. Later, Megiddo et al. [26] solved the problem in $O(n \log^2 n)$ time, and the algorithm was improved to $O(n \log n)$ time [17]. Finally, Frederickson [16] solved the problem in $O(n)$ time. The above four papers also solve the discrete case and the following problem version in the same running times: All points of T are considered as demand points and the centers are required to be at vertices of T. Further, if all points of T are demand points and centers can be any points of T, Megiddo and Tamir solved the problem in $O(n \log^3 n)$ time [25], and the running time can be reduced to $O(n \log^2 n)$ by applying Cole's parametric search [12].

As related problems, Frederickson [15] presented $O(n)$-time algorithms for the following tree partitioning problems: remove k edges from T such that the maximum (resp., minimum) total weight of all connected subtrees is minimized (resp., maximized).

Finding k centers in a general graph is NP-hard [20]. The geometric version of the problem in the plane is also NP-hard [24], i.e., finding k centers for n demanding points. Some special cases, however, are solvable in polynomial time. For example, if $k = 1$, then the problem can be solved in $O(n)$ time [23], and if $k = 2$, it can be solved in $O(n \log^2 n \log^2 \log n)$ time [7] (also refer to [1] for a faster randomized algorithm). If we require all centers to be on a given line, then the problem of finding k centers can be solved in polynomial time [5, 21, 28]. Recently, problems on uncertain data have been studied extensively and some k-center problem variations on uncertain data were also considered, e.g., [13, 18, 27, 29–31].

Our approach. We discuss our approach for the non-discrete problem, and that for the discrete case is similar (and even simpler). Let λ^* be the optimal objective value, i.e., $\lambda^* = \max_{v \in V(T)} \min_{q \in Q}\{w(v) \cdot d(v, q)\}$ for an optimal solution Q. A *feasibility test* is to determine whether $\lambda \geq \lambda^*$ for a given value λ, and if yes, we call λ a *feasible value*. Given any λ, the feasibility test can be done in $O(n)$ time [20].

Our algorithm follows an algorithmic scheme in [16] for the unweighted case, which is similar to that in [15] for the tree partition problems. However, a big difference is that three schemes were proposed in [15, 16] to gradually solve the problems in $O(n)$ time, while our approach only follows the first scheme and this significantly simplifies the algorithm. One reason the first scheme is sufficient to us is that our algorithm runs in $O(n \log n)$ time, which has a logarithmic factor more than the feasibility test algorithm. In contrast, most efforts of the last two schemes of [15, 16] are to reduce the running time of the algorithms to $O(n)$, which is the same as their corresponding feasibility test algorithms.

More specifically, our algorithm consists of two phases. The first phase will gather information so that each feasibility test can be done faster in sub-linear time. By using the

faster feasibility test algorithm, the second phase computes the optimal objective value λ^*. As in [16], we also use a stem-partition of the tree T. In addition to a matrix searching algorithm [14], we utilize some other techniques, such as 2D sublist LP queries [10] and line arrangements searching [11], etc.

In the following, in Section 2, we review some previous techniques that will be used later in our algorithm. In Section 3, we describe our techniques for dealing with a so-called "stem". We finally solve the k-center problem on T in Section 4. Due to the space limit, many details (including the algorithm for the discrete case) are omitted but can be found in the full paper.

2 Preliminaries

2.1 The feasibility test FTEST0

Given any value λ, the feasibility test is to determine whether λ is feasible, i.e., whether $\lambda \geq \lambda^*$. We say that a vertex v of T is *covered* (under λ) by a center q if $w(v) \cdot d(v,q) \leq \lambda$. Note that λ is feasible if and only if we can place k centers in T such that all vertices are covered. In the following we describe a linear-time feasibility test algorithm, which is essentially the same as the one in [20] although our description is much simpler.

We pick a vertex of T as the root, denoted by γ. For each vertex v, we use $T(v)$ to denote the subtree of T rooted at v. Following a post-order traversal on T, we place centers in a bottom-up and greedy manner. For each vertex v, we maintain two values $sup(v)$ and $dem(v)$, where $sup(v)$ is the distance from v to the closest center that has been placed in $T(v)$, and $dem(v)$ is the maximum distance from v such that if we place a center q within such a distance from v then all uncovered vertices of $T(v)$ can be covered by q. We also maintain a variable *count* to record the number of centers that have been placed so far.

Initially, $count = 0$, and for each vertex v, $sup(v) = \infty$ and $dem(v) = \frac{\lambda}{w(v)}$. Following a post-order traversal on T, suppose vertex v is being visited. For each child u of v, we update $sup(v)$ and $dem(v)$ as follows. If $sup(u) \leq dem(u)$, then we can use the center of $T(u)$ closest to u to cover the uncovered vertices of $T(u)$, and thus we reset $sup(v) = \min\{sup(v), sup(u)+d(u,v)\}$. Note that since u connects v by an edge, $d(v,u)$ is the length of the edge. Otherwise, if $dem(u) < d(u,v)$, then we place a center on the edge $e(u,v)$ at distance $dem(u)$ from u, so we update $count = count + 1$ and $sup(v) = \min\{sup(v), d(u,v) - dem(u)\}$. Otherwise (i.e., $dem(u) \geq d(u,v)$), we update $dem(v) = \min\{dem(v), dem(u) - d(u,v)\}$.

After the root γ is visited, if $sup(\gamma) > dem(\gamma)$, then we place a center at γ and update $count = count + 1$. Finally, λ is feasible if and only if $count \leq k$. The algorithm runs in $O(n)$ time. We use *FTEST0* to refer to the algorithm. To solve the k-center problem, the key is to compute λ^*, after which we can find k centers by applying *FTEST0* with $\lambda = \lambda^*$.

Remark. The algorithm *FTEST0* actually partitions T into at most k disjoint connected subtrees such that the vertices in each subtree is covered by the same center that is located in the subtree. We will make use of this observation later.

2.2 A matrix searching algorithm

We review an algorithm *MSEARCH*, which was proposed in [14] and was widely used, e.g., [15–17]. A matrix is *sorted* if elements in every row and every column are in nonincreasing order. Given a set of sorted matrices, a searching range (λ_1, λ_2) such that λ_2 is feasible and λ_1 is not, and a stopping count c, *MSEARCH* will produce a sequence of values one at a time for feasibility tests, and after each test, some elements in the matrices will be discarded.

Suppose a value λ is produced. If $\lambda \notin (\lambda_1, \lambda_2)$, we do not need to test λ. If λ is feasible, then λ_2 is updated to λ; otherwise, λ_1 is updated to λ. *MSEARCH* will stop once the number of remaining elements in all matrices is at most c. Lemma 1 is proved in [14] and we slightly change the statement to accommodate our need.

▶ **Lemma 1** ([14–17]). *Let $\mathcal{M}$ be a set of N sorted matrices $\{M_1, M_2, \ldots, M_N\}$ such that M_j is of dimension $m_j \times n_j$ with $m_j \leq n_j$, and $\sum_{j=1}^{N} m_j = m$. Let $c \geq 0$. The number of feasibility tests needed by* MSEARCH *to discard all but at most c of the elements is $O(\max\{\log \max_j\{n_j\}, \log(\frac{m}{c+1})\})$, and the total time of* MSEARCH *exclusive of feasibility tests is $O(\kappa \cdot \sum_{j=1}^{N} m_j \log(\frac{2n_j}{m_j}))$, where $O(\kappa)$ is the time for evaluating each matrix element (i.e., the number of matrix elements that need to be evaluated is $O(\sum_{j=1}^{N} m_j \log(\frac{2n_j}{m_j}))$).*

2.3 The 2D sublist LP queries

Let $H = \{h_1, h_2, \ldots, h_m\}$ be a set of m upper half-planes in the plane. Given two indices i and j with $1 \leq i \leq j \leq m$, a *2D sublist LP query* asks for the lowest point in the common intersection of $h_i, h_{i+1}, \ldots, h_j$. The *line-constrained* version of the query is: Given a vertical line l and two indices i and j with $1 \leq i \leq j \leq m$, the query asks for the lowest point on l in the common intersection of $h_i, h_{i+1}, \ldots, h_j$. Lemma 2 was proved in [10] (i.e., Lemma 8 and the discussion after it; the query algorithm for the line-constrained version is used as a procedure in the proof of Lemma 8).

▶ **Lemma 2** ([10]). *We can build a data structure for H in $O(m \log m)$ time such that each 2D sublist LP query can be answered in $O(\log^2 m)$ time, and each line-constrained query can be answered in $O(\log m)$ time.*

2.4 Line arrangement searching

Let L be a set of m lines in the plane. Denote by $\mathcal{A}(L)$ the arrangement of the lines of L, and let $y(v)$ be the y-coordinate of each vertex v of $\mathcal{A}(L)$. Let $v_1(L)$ be the lowest vertex of $\mathcal{A}(L)$ whose y-coordinate is a feasible value, and let $v_2(L)$ be the highest vertex of $\mathcal{A}(L)$ whose y-coordinate is smaller than that of $v_1(L)$. Hence, $y(v_2(L)) < \lambda^* \leq y(v_1(L))$, and $\mathcal{A}(L)$ does not have a vertex v with $y(v_2(L)) < y(v) < y(v_1(L))$. Lemma 3 was proved in [11].

▶ **Lemma 3** ([11]). *Both vertices $v_1(L)$ and $v_2(L)$ can be computed in $O((m+\tau)\log m)$ time, where τ is the time for a feasibility test.*

Remark. Alternatively, we can use Cole's parametric search [12] to compute the two vertices. First, we sort the lines of L by their intersections with the horizontal line $y = \lambda^*$, and this can be done in $O((m+\tau)\log m)$ time by Cole's parametric search [12]. Then, $v_1(L)$ and $v_2(L)$ can be found in additional $O(m)$ time because each of them is an intersection of two adjacent lines in the above sorted order. The line arrangement searching technique, which modified the slope selection algorithms [6, 22], avoids Cole's parametric search [12].

We will often talk about some problems in the plane $\mathbb{R}^2$, and if the context is clear, for any point $p \in \mathbb{R}^2$, we use $x(p)$ and $y(p)$ to denote its x- and y-coordinates, respectively.

3 The algorithms for stems

In this section, we first define *stems*, which are similar in spirit to those proposed in [16] for the unweighted case. Then, we will present two algorithms for solving the k-center problem on a stem, and both techniques will be used later for solving the problem in the tree T.

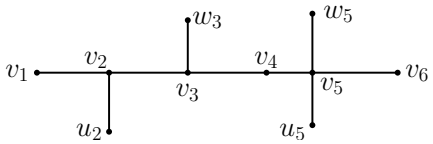


Figure 1 Illustrating a stem.

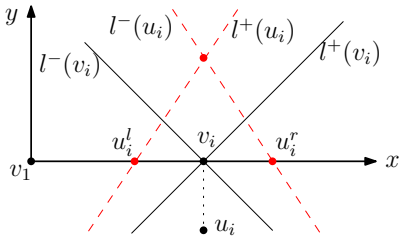


Figure 2 Illustrating the definitions of the lines defined by a backbone vertex v_i and its thorn vertex u_i.

Let $\widehat{P}$ be a path of m vertices, denoted by $v_1, v_2, \ldots, v_m$, sorted from left to right. For each vertex v_i, other than its incident edges in $\widehat{P}$, v_i has at most two additional edges connecting two vertices u_i and w_i that are not in $\widehat{P}$. Either vertex may not exist. Let P denote the union of $\widehat{P}$ and the above additional edges (e.g., see Fig. 1). For any two points p and q on P, we still use $\pi(p, q)$ to denote the unique path between p and q in P, and use $d(p, q)$ to denote the length of the path. With respect to a range (λ_1, λ_2), we call P a *stem* if the following holds: For each $i \in [1, m]$, if u_i exists, then $w(u_i) \cdot d(u_i, v_i) \leq \lambda_1$; if w_i exists, then $w(w_i) \cdot d(w_i, v_i) \geq \lambda_2$.

Following the terminology in [16], we call $e(v_i, u_i)$ a *thorn* and $e(v_i, w_i)$ a *twig*. Each u_i is called a *thorn vertex* and each w_i is called a *twig vertex*. $\widehat{P}$ is called the *backbone* of P, and each vertex of $\widehat{P}$ is called a *backbone vertex*. We define m as the *length* of P. The total number of vertices of P is at most $3m$.

Remark. Our algorithm in Section 4 will produce stems P as defined above, where $\widehat{P}$ is a path in T and all vertices of P are also vertices of T. However, each thorn $e(u_i, v_i)$ may not be an original edge of T, but it corresponds to the path between u_i and v_i in T in the sense that the length of $e(u_i, v_i)$ is equal to the distance between u_i and v_i in T. This is also the case for each twig $e(w_i, v_i)$. Our algorithm in Section 4 will maintain a range (λ_1, λ_2) such that λ_1 is not feasible and λ_2 is feasible, i.e., $\lambda^* \in (\lambda_1, \lambda_2]$. Since any feasibility test will be made to a value $\lambda \in (\lambda_1, \lambda_2)$, the above definitions on thorns and twigs imply the following: For each thorn vertex u_i, we can place a center on the backbone to cover it (under λ), and for each twig vertex w_i, we need to place a center on the edge $e(w_i, v_i) \setminus \{v_i\}$ to cover it.

In the sequel we give two different techniques for solving the k-center problem on the stem P. In fact, in our algorithm for the k-center problem on T in Section 4, we use these techniques to process a stem P, rather than directly solve the k-center problem on P. Let λ^* temporarily refer to the optimal objective value of the k-center problem on P in the rest of this section, and we assume $\lambda^* \in (\lambda_1, \lambda_2]$.

3.1 The first algorithm

This algorithm is motivated by the following easy observation: there exist two vertices v and v' in P such that a center q is located in the path $\pi(v, v')$ and $w(v) \cdot d(q, v) = w(v') \cdot d(q, v') = \lambda^*$.

We assume that all backbone vertices of P are in the x-axis of an xy-coordinate system $\mathbb{R}^2$ where v_1 is at the origin and each v_i has x-coordinate $d(v_1, v_i)$. Each v_i defines two lines $l^+(v_i)$ and $l^-(v_i)$ both containing v_i and with slopes $w(v_i)$ and $-w(v_i)$, respectively (e.g., see Fig. 2). Each thorn u_i also defines two lines $l^+(u_i)$ and $l^-(u_i)$ as follows. Define u_i^l (resp., u_i^r) to be the point in $\mathbb{R}^2$ on the x-axis with x-coordinate $d(v_1, v_i) - d(u_i, v_i)$ (resp., $d(v_1, v_i) + d(u_i, v_i)$). Hence, u_i^l (resp., u_i^r) is to the left (resp., right) of v_i with distance $d(u_i, v_i)$ from v_i. Define $l^+(u_i)$ to be the line through u_i^l with slope $w(u_i)$ and $l^-(u_i)$ to be the

line through u_i^r with slope $-w(u_i)$. Note that $l^+(u_i)$ and $l^-(u_i)$ intersect at the point whose x-coordinate is the same as that of v_i and whose y-coordinate is equal to $w(u_i) \cdot d(u_i, v_i)$. For each twig vertex w_i, we define points w_i^l and w_i^r, and lines $l^+(w_i)$ and $l^-(w_i)$, in the same way as those for u_i.

Consider a point q on the backbone of P to the right side of v_i. It can be verified that the weighted distance $w(v_i) \cdot d(v_i, q)$ from v_i to q is exactly equal to the y-coordinate of the intersection between $l^+(v_i)$ and the vertical line through q. If q is on the left side of v_i, we have a similar observation for $l^-(v_i)$. This is also true for u_i and w_i.

Let L denote the set of the lines in $\mathbb{R}^2$ defined by all vertices of P. Note that $|L| \leq 6m$. Based on the above observation, λ^* is equal to the y-coordinate of a vertex of the line arrangement $\mathcal{A}(L)$ of L. More precisely, λ^* is equal to the y-coordinate of $v_1(L)$, as defined in Section 2. By Lemma 3, we can compute λ^* in $O((m+\tau)\log m)$ time.

3.2 The second algorithm

This algorithm relies on the algorithm *MSEARCH*. We first form a set of sorted matrices.

For each $i \in [1, m]$, we define the two lines $l_i^+(v_i)$ and $l_i^-(v_i)$ in $\mathbb{R}^2$ as above in Section 3.1. If u_i exists, then we also define $l_i^+(u_i)$ and $l_i^-(u_i)$ as before; otherwise, both $l_i^+(u_i)$ and $l_i^-(u_i)$ refer to the x-axis. Let $h_{4(i-1)+j}$, $1 \leq j \leq 4$, denote respectively the four upper half-planes bounded by the above four lines (their index order is arbitrary). In this way, we have a set $H = \{h_1, h_2 \ldots, h_{4m}\}$ of $4m$ upper half-planes.

For any i and j with $1 \leq i \leq j \leq m$, we define $\alpha(i, j)$ as the y-coordinate of the lowest point in the common intersection of the upper half-planes of H from $h_{4(i-1)+1}$ to h_{4j}, i.e., all upper half-planes defined by u_t and v_t for $t \in [i, j]$. Observe that if we use one center to cover all backbone and thorn vertices u_t and v_t for $t \in [i, j]$, then $\alpha(i, j)$ is equal to the optimal objective value of this one-center problem.

We define a matrix M of dimension $m \times m$: For any i and j in $[1, m]$, if $i + j \leq m + 1$, then $M[i, j] = \alpha[i, m + 1 - j]$; otherwise, $M[i, j] = 0$.

For each twig w_i, we define two arrays A_i^r and A_i^l of at most m elements each as follows. Let $h^+(w_i)$ and $h^-(w_i)$ denote respectively the upper half-planes bounded by the lines $l^+(w_i)$ and $l^-(w_i)$ defined in Section 3.1. The array A_i^r is defined on the vertices of P on the right side of v_i, as follows. For each $j \in [1, m - i + 1]$, if we use a single center to cover w_i and all vertices u_t and v_t for $t \in [i, m + 1 - j]$, then $A_i^r[j]$ is defined to be the optimal objective value of this one-center problem, which is equal to the y-coordinate of the lowest point in the common intersection of $h^+(w_i)$ and the upper half-planes of H from $h_{4(i-1)+1}$ to $h_{4(m+1-j)}$. Symmetrically, array A_i^l is defined on the left side of v_i. Specifically, for each $j \in [1, i]$, if we use one center to cover w_i and all vertices u_t and v_t for $t \in [j, i]$, then $A^l[j]$ is defined to be the optimal objective value, which is equal to the y-coordinate of the lowest point in the common intersection of $h^-(w_i)$ and the upper half-planes of H from $h_{4(j-1)+1}$ to h_{4i}.

Let $\mathcal{M}$ be the set of the matrices M and A_i^r and A_i^l for all $1 \leq i \leq m$. The following lemma implies that we can apply *MSEARCH* on $\mathcal{M}$ to compute λ^*.

▶ **Lemma 4.** *Each matrix of $\mathcal{M}$ is sorted, and λ^* is an element of a matrix in $\mathcal{M}$.*

Proof. Refer to the full paper for the proof that every matrix of $\mathcal{M}$ is sorted. In the following, we show that λ^* must be an element of one of these matrices. We only sketch the proof.

Imagine that we apply algorithm *FTEST0* on $\lambda = \lambda^*$ and the stem P by considering P as a tree with root v_m. Then, *FTEST0* will compute at most k centers in P. *FTEST0* actually partitions P into at most k disjoint connected subtrees such that the vertices in each subtree is covered by the same center that is located in the subtree. Further, there must be a subtree

P_1 that has a center q and two vertices v' and v such that $w(v) \cdot d(v, q) = w(v') \cdot d(v', q) = \lambda^*$. Since P_1 is connected and both v and v' are in P_1, the path $\pi(v, v')$ is also in P_1.

Depending on whether one of v and v' is a twig vertex, there are two cases.

If neither vertex is a twig vertex, then we claim that all thorn vertices connecting to the backbone vertices of $\pi(v, v')$ are covered by the center q (see the full paper for the proof of the claim). If v is a backbone vertex, then let i be its index, i.e., $v = v_i$; otherwise, v is a thorn vertex and let i be the index such that v connects the backbone vertex v_i. Similarly, define j for v'. Without loss of generality, assume $i \leq j$. The above claim implies that λ^* is equal to the y-coordinate of the lowest point in the common intersection of all upper half-planes defined by the backbone vertices v_t and thorn vertices u_t for all $t \in [i, j]$, and thus, $\lambda^* = \alpha(i, j)$, which is equal to $M[i, m + 1 - j]$. Therefore, λ^* is in the matrix M.

Next, we consider the case where at least one of v and v' is a twig vertex. For each twig vertex w_i of P, by definition, $w(w_i) \cdot d(w_i, v_i) \geq \lambda_2$, and since $\lambda^* \leq \lambda_2$, the twig $e(w_i, v_i)$ must contain a center. Because both v and v' are covered by q, only one of them is a twig vertex. Without loss of generality, we assume that v is a twig vertex, say, w_i. If v' is a backbone vertex, then let j be its index; otherwise, v' is a thorn vertex and let j be the index such that v' connects the backbone vertex v_j. Without loss of generality, we assume $i \leq j$.

By the same argument as the above, all thorn vertices u_t with $t \in [i, j]$ are covered by q. This implies that λ^* is the y-coordinate of the lowest point in the common intersection of $h^+(w_i)$ and all upper half-planes defined by the backbone vertices v_t and thorn vertices u_t for all $t \in [i, j]$. Thus, $\lambda^* = A_i^r[m + 1 - j]$. Therefore, λ^* is in the array A_i^r. ◀

Note that $\mathcal{M}$ consists of a matrix M of dimension $m \times m$ and $2m$ arrays of lengths at most m. With the help of the 2D sublist LP query data structure in Lemma 2, the following lemma shows that the matrices of $\mathcal{M}$ can be implicitly formed in $O(m \log m)$ time.

▶ **Lemma 5.** *With $O(m \log m)$ time preprocessing, each matrix element of $\mathcal{M}$ can be evaluated in $O(\log^2 m)$ time.*

Proof. We build a data structure of Lemma 2 on the upper half-planes of H in $O(m \log m)$ time. Then, each element of M can be obtained in $O(\log^2 m)$ time by a 2D sublist LP query.

Now consider an array A_i^l. Given any index j, to compute $A_i^l[j]$, recall that $A_i^l[j]$ is equal to the y-coordinate of the lowest point p^* of the common intersection of the upper half-plane $h^+(w_i)$ and those in H', where H' is the set of the upper half-planes of H from $h_{4(j-1)+1}$ to h_{4i}. The lowest point p' of the common intersection of the upper half-planes of H' can be computed in $O(\log^2 m)$ time by a 2D sublist LP query with query indices $4(j-1)+1$ and $4i$. Computing p^* can also be done in $O(\log^2 m)$ time by slightly modifying the query algorithm for computing p'. We briefly discuss it below and the interested reader should refer to [10] for details (the proof of Lemma 8 and the discussion after the lemma).

The query algorithm for computing p' is similar in spirit to the linear-time algorithm for the 2D linear programming problem in [23]. It is a binary search algorithm. In each iteration, the algorithm computes the highest intersection p'' between a vertical line l and the bounding lines of the half-planes of H', and based on the local information at the intersection, the algorithm will determine which side to proceed for the search. For computing p^*, we need to incorporate the additional half-plane $h^+(w_i)$. To this end, in each iteration of the binary search, after we compute the highest intersection p'', we compare it with the intersection of l and the bounding line of $h^+(w_i)$ and update the highest intersection if needed. This costs only constant extra time for each iteration. Therefore, computing p^* takes $O(\log^2 m)$ time.

Computing the elements of arrays A_i^r can be done similarly. The lemma thus follows. ◀

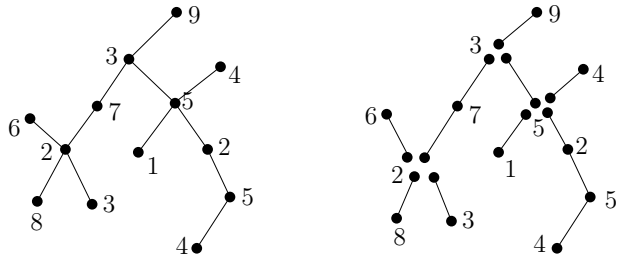


Figure 3 Left: the tree T where the numbers are the vertex weights. Right: the path partition of T.

By applying algorithm *MSEARCH* on $\mathcal{M}$ with stopping count $c = 0$ and $\kappa = O(\log^2 m)$, according to Lemma 1, *MSEARCH* produces $O(\log m)$ values for feasibility tests, and the total time exclusive of feasibility tests is $O(m \log^3 m)$ because we need to evaluate $O(m \log m)$ matrix elements of $\mathcal{M}$. Hence, the total time for computing λ^* is $O(m \log^3 m + \tau \cdot \log m)$.

Remark. Clearly, the first algorithm is better than the second one. However, later when we use the techniques of the second algorithm, m is often bounded by $O(\log^2 n)$ and thus $\log^3 m = O(\log n)$. In fact, we use the techniques of the second algorithm mainly because we need to set the stopping count c to some non-zero value.

4 Solving the k-center problem on T

In this section, we present our algorithm for solving the k-center problem on T. We will focus on computing the optimal objective value λ^*. Frederickson [15] proposed a *path-partition* of T, which is a partition of the edges of T into paths where a vertex v is an endpoint of a path if and only if the degree of v in T is not equal to 2 (e.g., see Fig. 3). A path in a partition-partition of T is called a *leaf-path* if it contains a leaf of T.

As in [16], we generalize the path-partition to stem-partition as follows. During the course of our algorithm, a range $(\lambda_1, \lambda_2]$ that contains λ^* will be maintained and T will be modified by removing some edges and adding some thorns and twigs. At any point in our algorithm, let T' be T with all thorns and twigs removed. A *stem* of T is a path in the path-partition of T', along with all thorns and twigs that connect to vertices in the path. A *stem-partition* of T is to partition T into stems according to a path-partition of T'. A stem in a stem-partition of T is called a *leaf-stem* if it contains a leaf of T that is a backbone vertex of the stem.

Our algorithm follows the first algorithmic scheme in [16]. There are two main phases: Phase 1 and Phase 2. Let $r = \log^2 n$. Phase 1 gathers information so that the feasibility test can be made in sublinear $O(\frac{n}{r} \log^3 r)$ time. Phase 2 computes λ^* by using the faster feasibility test. If T has more than $2n/r$ leaves, then there is an additional phase, called Phase 0, which reduces the problem to a tree with at most $2n/r$ leaves. In the following, we consider the general case where T has more than $2n/r$ leaves.

4.1 The preprocessing and computing the vertex ranks

We first perform some preprocessing. Recall that γ is the root of T. We compute the distances $d(v, \gamma)$ for all vertices v in $O(n)$ time. Then, if u is an ancestor of v, $d(u, v) = d(\gamma, v) - d(\gamma, u)$, which can be computed in $O(1)$ time. In the following, whenever we need to compute a distance $d(u, v)$, it is always the case that one of u and v is an ancestor of the other, and thus $d(u, v)$ can be obtained in $O(1)$ time.

Next, we compute a "rank" $rank(v)$ for each vertex v of T. These ranks will facilitate our algorithm later. For each vertex v, we define a point $p(v)$ on the x-axis with x-coordinate equal to $d(\gamma, v)$ in an xy-coordinate system $\mathbb{R}^2$, and define $l(v)$ as the line through $p(v)$ with slope equal to $-w(v)$. Let L be the set of these n lines. Consider the line arrangement $\mathcal{A}(L)$ of L. Let $v_1(L)$ and $v_2(L)$ be the vertices as defined in Section 2. By Lemma 3, both vertices can be computed in $O(n \log n)$ time. Let l be a horizontal line strictly between $v_1(L)$ and $v_2(L)$. We sort all lines of L by their intersections with l from left to right, and for each vertex v, we define $rank(v) = i$ if there are $i - 1$ lines before $l(v)$ in the above order. By the definitions of $v_1(L)$ and $v_2(L)$, the above order of L is also an order of L sorted by their intersections with the horizontal line $y = \lambda^*$.

4.2 Phase 0

Recall that T has more than $2n/r$ leaves. In this section, we reduce the problem to the problem of placing centers in a tree with at most $2n/r$ leaves. Our algorithm will maintain a range $(\lambda_1, \lambda_2]$ that contains λ^*. Initially, $\lambda_1 = y(v_2(L))$, the y-coordinate of $v_2(L)$, which is already computed in the preprocessing, and $\lambda_2 = y(v_1(L))$. We form a stem-partition of T, which is actually a path-partition since there are no thorns and twigs initially, and this can be done in $O(n)$ time.

Recall that $r = \log^2 n$. While there are more than $2n/r$ leaves in T, we do the following.

Recall that the length of a stem is defined as the number of backbone vertices. Let S be the set of all leaf-stems of T whose lengths are at most r. Let n' be the number of all backbone vertices on the leaf-stems of S. For each leaf-stem of S, we form matrices by Lemma 5. Let $\mathcal{M}$ denote the collection of matrices for all leaf-stems of S. We call *MSEARCH* on $\mathcal{M}$, with stopping count $c = n'/(2r)$, by using the feasibility test algorithm *FTEST0*. After *MSEARCH* stops, we have an updated range (λ_1, λ_2) and matrix elements of $\mathcal{M}$ in (λ_1, λ_2) are called *active* values. Since $c = n'/(2r)$, at most $n'/(2r)$ active values of $\mathcal{M}$ remain, and thus at most $n'/(2r)$ leaf-stems of S have active values.

For each leaf-stem $P \in S$ without active values, we perform the following *post-processing procedure*. The backbone vertex of P closest to the root is called the *top vertex*. We place centers on P, subtract their number from k, and replace P by either a thorn or a twig connected to the top vertex (P is thus removed from T except the top vertex), such that solving the k-center problem on the modified T also solves the problem on the original T. The post-processing procedure can be implemented in $O(m)$ time, where m is the length of P. The details are given below.

The post-processing procedure on P. Let z be the top vertex of P. We run *FTEST0* on P with z as the root and $\lambda = \lambda'$ that is an arbitrary value in (λ_1, λ_2). After z is finally processed, depending on whether $sup(z) \leq dem(z)$, we do the following.

If $sup(z) \leq dem(z)$, then let q be the last center that has been placed. In this case, all vertices of P are covered and z is covered by q. According to algorithm *FTEST0* and as discussed in the proof of Lemma 4, q covers a connected subtree of vertices, and let $V(q)$ denote the set of these vertices excluding z. Note that $V(q)$ can be easily identified during *FTEST0*. Let k' be the number of centers excluding q that have been placed on P. Since $\lambda' \in (\lambda_1, \lambda_2)$ and the matrices formed based on P do not have any active values, we have the following *key observation:* if we run *FTEST0* with any $\lambda \in (\lambda_1, \lambda_2)$, the algorithm will also cover all vertices of $P \setminus (V(q) \cup \{z\})$ with k' centers and cover vertices of $V(q) \cup \{z\}$ with one center. Indeed, this is true because the way we form matrices for P is consistent with *FTEST0*, as discussed in the proof of Lemma 4. In this case, we replace P by attaching a

twig $e(u, z)$ to z with length equal to $d(u, z)$, where u is a vertex of $V(q)$ with the following property: For any $\lambda \in (\lambda_1, \lambda_2)$, if we place a center q' on the path $\pi(u, z)$ at distance $\lambda/w(u)$ from u, then q' will cover all vertices of $V(q)$ under λ, i.e., u "dominates" all other vertices of $V(q)$ and thus it is sufficient to keep u (since λ_2 is feasible, any subsequent feasibility test in the algorithm will use $\lambda \in (\lambda_1, \lambda_2)$). The following lemma shows that u is the vertex of $V(q)$ with the largest rank. The proof of the lemma is omitted.

▶ **Lemma 6.** *Let u be the vertex of $V(q)$ with the largest rank. For any $\lambda \in (\lambda_1, \lambda_2)$, the following holds.*

1. $\frac{\lambda}{w(u)} \leq d(u, z)$.
2. *If q' is the point on the path $\pi(u, z)$ with distance $\frac{\lambda}{w(u)}$ from u, then q' covers all vertices of $V(q)$ under λ, i.e., $w(v) \cdot d(v, q') \leq \lambda$ for all $v \in V(q)$.*

Due to the preprocessing in Section 4.1, we can find u from $V(q)$ in $O(m)$ time. This finishes our post-processing procedure for the case $sup(z) \leq dem(z)$. Since $\frac{\lambda}{w(u)} \leq d(u, z)$ for any $\lambda \in (\lambda_1, \lambda_2)$, we have $w(u) \cdot d(u, z) \geq \lambda_2$, and thus, $e(u, z)$ is indeed a twig.

Next, we consider the other case $sup(z) > dem(z)$. In this case, P has some vertices other than z that are not covered yet, and we would need to place a center at z to cover them. Let V be the set of all uncovered vertices other than z, and V can be identified during *FTEST0*. In this case, we replace P by attaching a thorn $e(u, z)$ to z with length equal to $d(u, z)$, where u is a vertex of V with the following property: For any $\lambda \in (\lambda_1, \lambda_2)$, if there is a center q outside P covering u *through* z (by "through", we mean that $\pi(q, u)$ contains z) under distance λ, then q also covers all other vertices of V (intuitively u "dominates" all other vertices of V). Since later we will place centers outside P to cover the vertices of V through z under some $\lambda \in (\lambda_1, \lambda_2)$, it is sufficient to maintain u. The following lemma shows that u is the vertex of V with the largest rank. The proof is omitted.

▶ **Lemma 7.** *Let u be the vertex of V with the largest rank. Then, for any center q outside P that covers u through z under any $\lambda \in (\lambda_1, \lambda_2)$, q also covers all other vertices of V.*

Since a center at z would cover u, it holds that $w(u) \cdot d(u, z) \leq \lambda$ for any $\lambda \in (\lambda_1, \lambda_2)$, which implies that $w(u) \cdot d(u, z) \leq \lambda_1$. Thus, $e(u, z)$ is indeed a thorn.

The above replaces P by attaching to z either a thorn or a twig. We perform the following additional processing.

Suppose z is attached by a thorn $e(z, u)$. If z already has another thorn $e(z, u')$, then we discard one of u' and u whose rank is smaller, because any center that covers the remaining vertex will cover the discarded one as well. This makes sure that z has at most one thorn.

Suppose z is attached by a twig $e(z, u)$. If z already has another twig $e(z, u')$, then we can discard one of u and u' whose rank is *larger* (and subtract 1 from k). The reason is the following. Without loss of generality, assume $rank(u) < rank(u')$. Since both $e(z, u)$ and $e(z, u')$ are twigs, if we apply *FTEST0* on any $\lambda \in (\lambda_1, \lambda_2)$, then the algorithm will place a center q on $e(z, u)$ with distance $\lambda/w(u)$ from u and place a center q' on $e(z, u')$ with distance $\lambda/w(u')$ from u'. As $rank(u) < rank(u')$, we have Lemma 8.

▶ **Lemma 8.** $d(q, z) \leq d(q', z)$.

Lemma 8 tells that any vertex covered by q' in the subsequent algorithm will also be covered by q. Thus, it suffices to maintain the twig $e(z, u)$. Since we need to place a center at $e(z, u')$, we subtract 1 from k after removing $e(z, u')$. Hence, z has at most one twig.

This finishes the post-processing procedure for P. Due to the preprocessing in Section 4.1, the running time of the procedure is $O(m)$.

Let T be the modified tree after the post-processing on each stem P without active values. If T still has more than $2n/r$ leaves, then we repeat the above. The algorithm stops once T has at most $2n/r$ leaves. This finishes Phase 0. The following lemma gives the time analysis.

▶ **Lemma 9.** *Phase 0 runs in $O(n(\log\log n)^3)$ time.*

Proof. We first argue that the number of iterations of the while loop is $O(\log r)$.

We consider an iteration of the while loop. Suppose the number of leaf-stems in T, denoted by m, is at least $2n/r$. Then, at most n/r leaf-stems are of length larger than r. Hence, at least half of the leaf-stems are of length at most r. Thus, $|S| \geq m/2$. Recall that n' is the total number of backbone vertices in all leaf-stems of S. Because at most $n'/(2r)$ leaf-stems have active values after *MSEARCH*, at least $|S| - n'/(2r) \geq m/2 - n'/(2r) \geq m/2 - n/(2r) \geq m/2 - m/4 = m/4$ leaf-stems will be removed. Note that removing two such leaf-stems may make an interior vertex become a new leaf in the modified tree. Hence, the tree resulting at the end of each iteration will have at most 7/8 of the leaf-stems of the tree at the beginning of the iteration. Therefore, the number of iterations of the while loop needed to reduce the number of leaf-stems to at most $2n/r$ is $O(\log r)$.

We proceed to analyze the running time of Phase 0. In each iteration of the while loop, we call *MSEARCH* on the matrices for all leaf-stems of S. Since the length of each stem P of S is at most r, there are $O(r)$ matrices formed for P. We perform the preprocessing of Lemma 5 on the matrices, so that each matrix element can be evaluated in $O(\log^2 r)$ time. The total time of the preprocessing on stems of S is $O(n' \log r)$. Since $\mathcal{M}$ has $O(n')$ matrices and the stopping account c is $n'/(2r)$, each call to *MSEARCH* produces $O(\log r)$ values for feasibility tests in $O(n' \log^3 r)$ time (i.e., $O(n' \log r)$ matrix elements will be evaluated). For each leaf-stem without active values, the post-processing time for it is $O(r)$. Hence, the total post-processing time in each iteration is $O(n')$.

Since there are $O(\log r)$ iterations, the total number of feasibility tests is $O(\log^2 r)$, and thus the overall time for all feasibility tests in Phase 0 is $O(n \log^2 r)$. On the other hand, after each iteration, at most $n'/(2r)$ leaf-stems of S have active values and other leaf-stems of S will be deleted. Since the length of each leaf-stem of S is at most r, the leaf-stems with active values have at most $n'/2$ backbone vertices, and thus at least $n'/2$ backbone vertices will be deleted in each iteration. Therefore, the total sum of such n' in all iterations is $O(n)$. Hence, the total time for the preprocessing of Lemma 5 is $O(n \log r)$, the total time for *MSEARCH* is $O(n \log^3 r)$, and the total post-processing time for leaf-stems without active values is $O(n)$.

In summary, the overall time of Phase 0 (excluding the preprocessing in Section 4.1) is $O(n \log^3 r)$, which is $O(n(\log\log n)^3)$ since $r = \log^2 n$. ◀

4.3 Phase 1

We assume that the tree T now has at most $2n/r$ leaves and we want to place k centers in T to cover all vertices. Note that T may have some thorns and twigs. The main purpose of this phase is to gather information so that each feasibility test can be done in sublinear time, and specifically, $O(n/r \log^3 r)$ time. Recall that we have a range $(\lambda_1, \lambda_2]$ that contains λ^*.

We first form a stem-partition for T. Then, we further partition the stems into substems, each of length at most r, such that the lowest backbone vertex v in a substem is the highest backbone vertex in the next lower substem (if v has a thorn or/and a twig, then they are included in the upper substem). So this results in a partition of edges. Let S be the set of all substems. Let T_c be the tree in which each node represents a substem of S and node μ in T_c is the parent of node ν if the highest backbone vertex of the substem for ν is the lowest

backbone vertex of the substem for μ, and we call T_c the *stem tree*. As in [15, 16], since T has at most $2n/r$ leaves, $|S| = O(n/r)$ and the number of nodes of T_c is $O(n/r)$.

For each substem $P \in S$, we compute the set L_P of lines as in Section 3.1. Let L be the set of all the lines for all substems of S. We define the lines of L in the same xy-coordinate system $\mathbb{R}^2$. Clearly, $|L| = O(n)$. Consider the line arrangement $\mathcal{A}(L)$. Define vertices $v_1(L)$ and $v_2(L)$ of $\mathcal{A}(L)$ as in Section 2. With Lemma 3 and *FTEST0*, both vertices can be computed in $O(n \log n)$ time. We update $\lambda_1 = \max\{\lambda_1, y(v_2(L))\}$ and $\lambda_2 = \min\{\lambda_2, y(v_1(L))\}$. Hence, we still have $\lambda^* \in (\lambda_1, \lambda_2]$. We again call the values in (λ_1, λ_2) *active* values.

For each substem $P \in S$, observe that each element of the matrices formed based on P in Section 3.2 is equal to the y-coordinate of the intersection of two lines of L_P, and thus is equal to the y-coordinate of a vertex of $\mathcal{A}(L)$. By the definitions of $v_1(L)$ and $v_2(L)$, no matrix element of P is active.

In the future algorithm, we will only need to test feasibilities for values $\lambda \in (\lambda_1, \lambda_2)$. We compute a data structure on each substem P of S, so that it will help make the feasibility test faster. We have the following lemma and use *FTEST1* to denote the feasibility test algorithm in the lemma. The lemma proof is omitted.

▶ **Lemma 10.** *After $O(n \log n)$ time preprocessing, each feasibility test can be done in $O(n/r \cdot \log^3 r)$ time.*

4.4 Phase 2

In this phase, we will finally compute the optimal objective value λ^*, using the faster feasibility test *FTEST1*. Recall that we have computed a range $(\lambda_1, \lambda_2]$ that contains λ^* after Phase 1.

We first form a stem-partition of T. While there is more than one leaf-stem, we do the following. Let S be the set of all leaf-stems. For each stem $P \in S$, we compute the set of lines as in Section 3.1, and let L be the set of the lines for all stems of S. With Lemma 3 and *FTEST1*, we compute the two vertices $v_1(L)$ and $v_2(L)$ of the arrangement $\mathcal{A}(L)$ as defined in Section 2. We update $\lambda_1 = \max\{\lambda_1, y(v_2(L))\}$ and $\lambda_2 = \min\{\lambda_2, y(v_1(L))\}$. As discussed in Phase 1, each stem P of S does not have any active values (in the matrices defined by P). Next, for each stem P of S, we perform the post-processing procedure as in Section 4.2, i.e., place centers on P, subtract their number from k, and replace P by attaching a twig or a thorn to its top vertex. Let T be the modified tree.

After the while loop, T is a single stem. Then, we apply above algorithm on the only stem T, and the obtained value λ_2 is λ^*. The running time of Phase 2 is bounded by $O(n \log n)$, which is proved in the following theorem.

▶ **Theorem 11.** *The k-center problem on T can be solved in $O(n \log n)$ time.*

Proof. As discussed before, Phases 0 and 1 run in $O(n \log n)$ time. We focus on Phase 2.

First of all, as in [16], the number of iterations of the while loop is $O(\log n)$ because the number of leaf-stems is halved after each iteration. In each iteration, let n' denote the total number of backbone vertices of all leaf-stems in S. Hence, $|L| = O(n')$. Thus, the call to Lemma 3 with *FTEST1* takes $O((n' + n/r \cdot \log^3 r) \log n')$ time. The total time of the post-processing procedure for all leaf-stems of S is $O(n')$. Since all leaf-stems of S will be removed in the iteration, the total sum of all such n' is $O(n)$ in Phase 2. Therefore, the total time of the algorithm in Lemma 3 in Phase 2 is $O(n \log n + n/r \cdot \log^3 r \log^2 n)$, which is $O(n \log n)$ since $r = \log^2 n$. Also, the overall time for the post-processing procedure in Phase 2 is $O(n)$. Therefore, the total time of Phase 2 is $O(n \log n)$. ◀

The pseudocode in Algorithm 1 summarizes the overall algorithm.

Algorithm 1: The k-center algorithm.

Input: A tree T and an integer k
Output: The optimal objective values λ^* and k centers in T

Perform the preprocessing in Section 4.1 and compute the "ranks" for all vertices of T;
`/* Phase 0 */`
$r \leftarrow \log^2 n$;
Form a stem-partition of T;
while do
  there are more than $2n/r$ leaves in T
Let S be the set of all leaf-stems of lengths at most r;
Form the set $\mathcal{M}$ of matrices for all leaf-stems of S by Lemma 5;
Let n' be the total number of all backbone vertices of the leaf-stems of S;
Call *MSEARCH* on $\mathcal{M}$ with stopping count $c = n'/(2r)$, using *FTEST0*;
for do
  each leaf-stem P of S with no active values
Perform the post-processing on P, i.e., place centers on P, subtract their number from k, replace P by a thorn or a twig, and modify the stem-partition of T;
`/* Phase 1 */`
For a stem-partition of T, and for each stem, partition it into substems of lengths at most r;
Let S be the set of all substems, and form the stem-tree T_c;
Compute the set L of lines for all stems of S in the way discussed in Section 3.1;
Compute the two vertices $v_1(L)$ and $v_2(L)$ of $\mathcal{A}(L)$ by Lemma 3 and *FTEST0*, and update λ_1 and λ_2;
for do
  each substem P of S
Compute the data structure for the faster feasibility test *FTEST1*;
`/* Phase 2 */`
Form a stem-partition of T;
while do
  there is more than one leaf-stem in T
Compute the set L of lines for all stems of S in the way discussed in Section 3.1;
Compute the two vertices $v_1(L)$ and $v_2(L)$ of $\mathcal{A}(L)$ by Lemma 3 and *FTEST1*, and update λ_1 and λ_2;
for do
  each leaf-stem of S
Perform the post-processing on P, i.e., place centers on P, subtract their number from k, replace P by a thorn or a twig, and modify the stem-partition of T;
Compute the set L of the lines for the only leaf-stem T;
Compute the two vertices $v_1(L)$ and $v_2(L)$ of $\mathcal{A}(L)$ by Lemma 3 and *FTEST1*, and update λ_1 and λ_2;
$\lambda^* = \lambda_2$;
Apply *FTEST0* on $\lambda = \lambda^*$ to find k centers in the original tree T;

References

1. P.K. Agarwal and J.M. Phillips. An efficient algorithm for 2D Euclidean 2-center with outliers. In *Proceedings of the 16th Annual European Conference on Algorithms(ESA)*, pages 64–75, 2008.
2. M. Ajtai, J. Komlós, and E. Szemerédi. An $O(n \log n)$ sorting network. In *Proc. of the 15th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1–9, 1983.
3. A. Banik, B. Bhattacharya, S. Das, T. Kameda, and Z. Song. The p-center problem in tree networks revisited. In *Proc. of the 15th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, pages 6:1–6:15, 2016.
4. B. Bhattacharya and Q. Shi. Optimal algorithms for the weighted p-center problems on the real line for small p. In *Proc. of the 10th International Workshop on Algorithms and Data Structures*, pages 529–540, 2007.
5. P. Brass, C. Knauer, H.-S. Na, C.-S. Shin, and A. Vigneron. The aligned k-center problem. *International Journal of Computational Geometry and Applications*, 21:157–178, 2011.
6. H Brönnimann and B. Chazelle. Optimal slope selection via cuttings. *Computational Geometry: Theory and Applications*, 10(1):23–29, 1998.
7. T.M. Chan. More planar two-center algorithms. *Computational Geometry: Theory and Applications*, 13:189–198, 1999.
8. R. Chandrasekaran and A. Tamir. Polynomially bounded algorithms for locating p-centers on a tree. *Mathematical Programming*, 22(1):304–315, 1982.
9. D.Z. Chen, J. Li, and H. Wang. Efficient algorithms for the one-dimensional k-center problem. *Theoretical Computer Science*, 592:135–142, 2015.
10. D.Z. Chen and H. Wang. Approximating points by a piecewise linear function. *Algorithmica*, 88:682–713, 2013.
11. D.Z. Chen and H. Wang. A note on searching line arrangements and applications. *Information Processing Letters*, 113:518–521, 2013.
12. R. Cole. Slowing down sorting networks to obtain faster sorting algorithms. *Journal of the ACM*, 34(1):200–208, 1987.
13. G. Cormode and A. McGregor. Approximation algorithms for clustering uncertain data. In *Proc. of the 27t Symposium on Principles of Database Systems (PODS)*, pages 191–200, 2008.
14. G. Frederickson and D. Johnson. Generalized selection and ranking: Sorted matrices. *SIAM Journal on Computing*, 13(1):14–30, 1984.
15. G.N. Frederickson. Optimal algorithms for tree partitioning. In *Proc. of the 2nd Annual ACM-SIAM Symposium of Discrete Algorithms (SODA)*, pages 168–177, 1991.
16. G.N. Frederickson. Parametric search and locating supply centers in trees. In *Proc. of the 2nd International Workshop on Algorithms and Data Structures (WADS)*, pages 299–319, 1991.
17. G.N. Frederickson and D.B. Johnson. Finding kth paths and p-centers by generating and searching good data structures. *Journal of Algorithms*, 4(1):61–80, 1983.
18. L. Huang and J. Li. Stochastic k-center and j-flat-center problems. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 110–129, 2017.
19. M. Jeger and O. Kariv. Algorithms for finding P-centers on a weighted tree (for relatively small P). *Networks*, 15(3):381–389, 1985.
20. O. Kariv and S.L. Hakimi. An algorithmic approach to network location problems. I: The p-centers. *SIAM Journal on Applied Mathematics*, 37(3):513–538, 1979.
21. A. Karmakar, S. Das, S.C. Nandy, and B.K. Bhattacharya. Some variations on constrained minimum enclosing circle problem. *Journal of Combinatorial Optimization*, 25(2):176–190, 2013.

22 M. Katz and M. Sharir. Optimal slope selection via expanders. *Information Processing Letters*, 47(3):115–122, 1993.

23 N. Megiddo. Linear-time algorithms for linear programming in R^3 and related problems. *SIAM Journal on Computing*, 12(4):759–776, 1983.

24 N. Megiddo and K.J. Supowit. On the complexity of some common geometric location problems. *SIAM Journal on Comuting*, 13:182–196, 1984.

25 N. Megiddo and A. Tamir. New results on the complexity of p-centre problems. *SIAM Journal on Computing*, 12(4):751–758, 1983.

26 N. Megiddo, A. Tamir, E. Zemel, and R. Chandrasekaran. An $O(n \log^2 n)$ algorithm for the k-th longest path in a tree with applications to location problems. *SIAM Journal on Computing*, 10:328–337, 1981.

27 H. Wang and J. Zhang. One-dimensional k-center on uncertain data. *Theoretical Computer Science*, 602:114–124, 2015.

28 H. Wang and J. Zhang. Line-constrained k-median, k-means, and k-center problems in the plane. *International Journal of Computational Geometry and Applications*, 26:185–210, 2016.

29 H. Wang and J. Zhang. A note on computing the center of uncertain data on the real line. *Operations Research Letters*, 44:370–373, 2016.

30 H. Wang and J. Zhang. Computing the center of uncertain points on tree networks. *Algorithmica*, 609:32–48, 2017.

31 H. Wang and J. Zhang. Covering uncertain points in a tree. In *Proc. of the 15th Algorithms and Data Structures Symposium (WADS)*, pages 557–568, 2017.

New Bounds for Range Closest-Pair Problems

Jie Xue
Department of Computer Science & Engineering, University of Minnesota
Minneapolis, MN, USA
http://cs.umn.edu/~xuexx193
xuexx193@umn.edu

Yuan Li
Facebook Inc.
Seattle, WA, USA
lydxlx@fb.com

Saladi Rahul
Department of Computer Science, University of Illinois
Urbana, IL, USA
http://cs.umn.edu/~rahuls
saladi.rahul@gmail.com

Ravi Janardan
Department of Computer Science & Engineering, University of Minnesota
Minneapolis, MN, USA
http://cs.umn.edu/~janardan
janardan@umn.edu

Abstract

Given a dataset S of points in $\mathbb{R}^2$, the range closest-pair (RCP) problem aims to preprocess S into a data structure such that when a query range X is specified, the closest-pair in $S \cap X$ can be reported efficiently. The RCP problem can be viewed as a range-search version of the classical closest-pair problem, and finds applications in many areas. Due to its non-decomposability, the RCP problem is much more challenging than many traditional range-search problems. This paper revisits the RCP problem, and proposes new data structures for various query types including quadrants, strips, rectangles, and halfplanes. Both worst-case and average-case analyses (in the sense that the data points are drawn uniformly and independently from the unit square) are applied to these new data structures, which result in new bounds for the RCP problem. Some of the new bounds significantly improve the previous results, while the others are entirely new.

2012 ACM Subject Classification Theory of computation → Computational geometry, Theory of computation → Data structures design and analysis

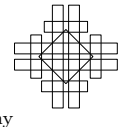
Keywords and phrases Closest-pair, Range search, Candidate pairs, Average-case analysis

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.73

Related Version A full version of this paper is available at [12], http://arxiv.org/abs/1712.09749.

1 Introduction

The closest-pair problem is one of the most fundamental problems in computational geometry and finds many applications, e.g., collision detection, similarity search, traffic control, etc. In this paper, we study a range-search version of the closest-pair problem called the *range closest-pair* (RCP) problem. Let $\mathcal{X}$ be a certain collection of ranges called *query space*. The

34th International Symposium on Computational Geometry (SoCG 2018).
Editors: Bettina Speckmann and Csaba D. Tóth; Article No. 73; pp. 73:1–73:14


Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

RCP problem with query space $\mathcal{X}$ (or the $\mathcal{X}$-RCP problem for short) aims to preprocess a given dataset S of points into a low-space data structure such that when a query range $X \in \mathcal{X}$ is specified, the closest-pair in $S \cap X$ can be reported efficiently. The motivation for the RCP problem is clear and similar to that of range search: in many situations, one is interested in local information (i.e., local closest-pairs) inside specified ranges rather than global information (i.e., global closest-pair) of the dataset.

The RCP problem is quite challenging due to a couple of reasons. First, in the RCP problem, the objects of interest are in fact point-pairs instead of single points, and in a dataset there is a quadratic number of point-pairs to be dealt with. Moreover, the RCP problem is non-decomposable in the sense that even if the query range $X \in \mathcal{X}$ can be written as $X = X_1 \cup X_2$, the closest-pair in $S \cap X$ cannot be computed from the closest-pairs in $S \cap X_1$ and $S \cap X_2$. The non-decomposability makes many traditional range-search techniques inapplicable to the RCP problem, and thus makes the problem much more challenging.

The RCP problem in $\mathbb{R}^2$ has been studied in prior work over the last fifteen years, e.g., [1, 5, 6, 9, 10]. In this paper, we revisit this problem and make significant improvements to the existing solutions. Following the existing work, the query types considered in this paper are orthogonal queries (specifically, quadrants, strips, rectangles) and halfplane query.

1.1 Our contributions, techniques, and related work

The closest-pair problem and range search are both classical topics in computational geometry; see [2, 11] for references. The RCP problem is relatively new. The best existing bounds in $\mathbb{R}^2$ and our new results are summarized in Table 1 (Space refers to space cost and Qtime refers to query time), and we give a brief explanation below.

Table 1 Summary of the best existing bounds and our new results for the RCP problem in $\mathbb{R}^2$ (each row corresponds to an RCP data structure for the corresponding query space).

Query	Source	Worst-case		Average-case	
		Space	Qtime	Space	Qtime
Quadrant	[6]	$O(n \log n)$	$O(\log n)$	-	-
	Theorem 3	$\boldsymbol{O(n)}$	$\boldsymbol{O(\log n)}$	$\boldsymbol{O(\log^2 n)}$	$\boldsymbol{O(\log \log n)}$
Strip	[10]	$O(n \log^2 n)$	$O(\log n)$	-	-
	Theorem 6	$\boldsymbol{O(n \log n)}$	$\boldsymbol{O(\log n)}$	$\boldsymbol{O(n)}$	$\boldsymbol{O(\log n)}$
Rectangle	[6]	$O(n \log^5 n)$	$O(\log^2 n)$	-	-
	[10]	$O(n \log^3 n)$	$O(\log^3 n)$	-	-
	[6]	-	-	$O(n \log^4 n)$	$O(\log^4 n)$
	Theorem 15	$\boldsymbol{O(n \log^2 n)}$	$\boldsymbol{O(\log^2 n)}$	$\boldsymbol{O(n \log n)}$	$\boldsymbol{O(\log n)}$
Halfplane	[1]	$O(n \log n)$	$O(n^{0.5+\varepsilon})$	-	-
	Theorem 18	$\boldsymbol{O(n)}$	$\boldsymbol{O(\log n)}$	$\boldsymbol{O(\log^2 n)}$	$\boldsymbol{O(\log \log n)}$

Related work. The RCP problem for orthogonal queries was studied in [6, 9, 10]. The best known solution for quadrant query was given by [6], while [10] gave the best known solution for strip query. For rectangle query, there are two best known solutions (in terms of worst-case bounds) given by [6] and [10] respectively. The above results only considered worst-case performance of the data structures. The authors of [6] for the first time applied average-case analysis to RCP data structures in the model where the data points are drawn independently and uniformly from the unit square. Unfortunately, [6] only gave a rectangle

RCP data structure with low average-case *preprocessing* time, while its average-case space cost and query time are even higher than the worst-case counterparts of the data structure given by [10] (even worse, its worst-case space cost is super-quadratic). In fact, in terms of space cost and query time, no nontrivial average-case bounds were known for any kind of query before this paper. The RCP problem for halfplane query was studied in [1]. Two data structures were proposed. We only present the first one in Table 1. The second one (not in the table), while having higher space cost and query time than the first one, can be built in $O(n \log^2 n)$ time. Both data structures require (worst-case) super-linear space cost and polynomial query time.

Our contributions. In this paper, we improve all the above results by giving new RCP data structures for various query types. The improvements can be seen in Table 1. In terms of worst-case bounds, the highlights are our rectangle RCP data structure which simultaneously improves the two best known results (given by [6] and [10]) and our halfplane RCP data structure which is *optimal* and significantly improves the bounds in [1]. Furthermore, by applying average-case analysis to our new data structures, we establish the first nontrivial average-case bounds for all the query types studied. Our average-case analysis applies to datasets generated in not only the unit square but also an arbitrary axis-parallel rectangle. These average-case bounds demonstrate that our new data structures might have much better performance in practice than one can expect from the worst-case bounds. Finally, we also give an $O(n \log^2 n)$-time algorithm to build our halfplane RCP data structure, matching the preprocessing time in [1]. The preprocessing for our orthogonal RCP data structures is not considered in this paper; we are still in the process of investigating this.

Our techniques. An important notion in our techniques is that of a *candidate pair*, i.e., a pair of data points that is the answer to some RCP query. Our solutions for the quadrant and strip RCP problems use the candidate pairs to construct a planar subdivision and take advantage of point-location techniques to answer queries. The data structures themselves are simple, and our main technical contribution here occurs in the average-case analysis of the data structures. The analysis requires a study of the expected number of candidate pairs in a random dataset, which is of both geometric and combinatorial interest. Our data structure for the rectangle RCP problem is subtle; it is constructed by properly combining two simpler data structures, each of which partially achieves the desired bounds. The high-level framework of the two simpler data structures is identical: it first "decomposes" a rectangle query into four quadrant queries and then simplifies the problem via some geometric observations similar to those in the standard divide-and-conquer algorithm for the classical closest-pair problem. Also, the analysis of the data structures is technically interesting. Our solution for the halfplane RCP problem applies the duality technique to map the candidate pairs to wedges in the dual space and form a planar subdivision, which allows us to solve the problem by using point-location techniques on the subdivision, similarly to the approach for the quadrant and strip RCP problems. However, unlike the quadrant and strip cases, to bound the complexity of the subdivision here is much more challenging, which requires non-obvious observations using the properties of duality and the problem itself. The average-case bounds of the data structure follow from a technical result bounding the expected number of candidate pairs, which also involves a nontrivial proof.

Organization. Section 1.2 presents the notations and preliminaries that are used throughout the paper. We suggest that the readers read this section carefully before moving on. Our solutions for quadrant, strip, rectangle, and halfplane queries are presented in Section 2, 3, 4,

and 5, respectively. In Section 6, we conclude our results and give some open questions for future work. Due to space limitations, proofs are omitted in this paper and can be found in the full version [12]. For the convenience of the reader, for some technical lemmas and theorems, we give short proof sketches in this paper which provide an overview of the proofs. Also, the details of the preprocessing algorithm for our halfplane RCP data structure is presented in [12].

1.2 Notations and Preliminaries

We introduce the notations and preliminaries that are used throughout the paper.

Query spaces. The following notations denote various query spaces (i.e., collections of ranges in $\mathbb{R}^2$): $\mathcal{Q}$ quadrants, $\mathcal{P}$ strips, $\mathcal{U}$ 3-sided rectangles, $\mathcal{R}$ rectangles, $\mathcal{H}$ halfplanes (quadrants, strips, 3-sided rectangles, rectangles under consideration are all axis-parallel). Define $\mathcal{Q}^{\nearrow} = \{[x,\infty)\times[y,\infty) : x,y\in\mathbb{R}\}\subseteq\mathcal{Q}$ as the sub-collection of all northeast quadrants, and define $\mathcal{Q}^{\nwarrow}, \mathcal{Q}^{\searrow}, \mathcal{Q}^{\swarrow}$ similarly. Define $\mathcal{P}^{\mathrm{v}} = \{[x_1,x_2]\times\mathbb{R} : x_1,x_2\in\mathbb{R}\}\subseteq\mathcal{P}$ as the sub-collection of all vertical strips, and similarly $\mathcal{P}^{\mathrm{h}}$ horizontal strips. If l is a vertical (resp., horizontal) line, an *l-anchored* strip is a vertical (resp., horizontal) strip containing l; define $\mathcal{P}_l\subseteq\mathcal{P}$ as the sub-collection of all l-anchored strips. Define $\mathcal{U}^{\downarrow} = \{[x_1,x_2]\times(-\infty,y] : x_1,x_2,y\in\mathbb{R}\}\subseteq\mathcal{U}$ as the sub-collection of all bottom-unbounded 3-sided rectangles, and define $\mathcal{U}^{\uparrow}$,$\mathcal{U}^{\leftarrow}$,$\mathcal{U}^{\rightarrow}$ similarly. If l is a non-vertical line, denote by $l^{\uparrow}$ (resp., $l^{\downarrow}$) the halfplane above (resp., below) l; define $\mathcal{H}^{\uparrow} = \{l^{\uparrow} : l \text{ is a non-vertical line}\}\subseteq\mathcal{H}$ (resp., $\mathcal{H}^{\downarrow} = \{l^{\downarrow} : l \text{ is a non-vertical line}\}\subseteq\mathcal{H}$).

Candidate pairs. For a dataset S and query space $\mathcal{X}$, a *candidate pair* of S with respect to $\mathcal{X}$ refers to a pair of points in S which is the closest-pair in $S\cap X$ for some $X\in\mathcal{X}$. We denote by $\Phi(S,\mathcal{X})$ the set of the candidate pairs of S with respect to $\mathcal{X}$. If l is a line, we define $\Phi_l(S,\mathcal{X})\subseteq\Phi(S,\mathcal{X})$ as the subset consisting of the candidate pairs that cross l (i.e., whose two points are on opposite sides of l).

Data structures. For a data structure $\mathcal{D}$, we denote by $\mathcal{D}(S)$ the data structure instance of $\mathcal{D}$ built on the dataset S. The notations $\mathsf{Space}(\mathcal{D}(S))$ and $\mathsf{Qtime}(\mathcal{D}(S))$ denote the space cost and query time (i.e., the maximum time for answering a query) of $\mathcal{D}(S)$, respectively.

Random datasets. If X is a region in $\mathbb{R}^2$ (or more generally in $\mathbb{R}^d$), we write $S\propto X^n$ to mean that S is a dataset of n random points drawn independently from the uniform distribution $\mathsf{Uni}(X)$ on X. More generally, if $X_1,\ldots,X_n$ are regions in $\mathbb{R}^2$ (or more generally in $\mathbb{R}^d$), we write $S\propto\prod_{i=1}^n X_i$ to mean that S is a dataset of n random points drawn independently from $\mathsf{Uni}(X_1),\ldots,\mathsf{Uni}(X_n)$ respectively.

Other notions. For a point $a\in\mathbb{R}^2$, we denote by $a.x$ and $a.y$ the x-coordinate and y-coordinate of a, respectively. For $a,b\in\mathbb{R}^d$, we use $\mathrm{dist}(a,b)$ to denote the Euclidean distance between a and b, and use $[a,b]$ to denote the segments connecting a and b (in $\mathbb{R}^1$ this coincides with the notation for a closed interval). We say $I_1,\ldots,I_n$ are vertical (resp., horizontal) *aligned* segments in $\mathbb{R}^2$ if there exist $r_1,\ldots,r_n,\alpha,\beta\in\mathbb{R}$ such that $I_i = \{r_i\}\times[\alpha,\beta]$ (resp., $I_i = [\alpha,\beta]\times\{r_i\}$). The *length* of a pair $\phi=(a,b)$ of points is the length of the segment $[a,b]$. For $S\subseteq\mathbb{R}^2$ of size at least 2, the notation $\kappa(S)$ denotes the *closest-pair distance* of S, i.e., the length of the closest-pair in S.

The following result regarding the closest-pair distance of a random dataset will be used to bound the expected number of candidate pairs with respect to various query spaces.

▶ **Lemma 1.** *Let R be a rectangle of size $\Delta \times \Delta'$ where $\Delta \leq \Delta'$, and $A \propto R^m$. Then*

$$\mathbb{E}[\kappa^p(A)] = \Theta\left(\max\left\{(\Delta'/m^2)^p, (\sqrt{\Delta\Delta'}/m)^p\right\}\right) \text{ for any constant } p > 1.$$

In particular, if R is a segment of length ℓ, then $\mathbb{E}[\kappa^p(A)] = \Theta((\ell/m^2)^p)$.

2 Quadrant query

We consider the RCP problem for quadrant queries, i.e., the $\mathcal{Q}$-RCP problem. In order to solve the $\mathcal{Q}$-RCP problem, it suffices to consider the $\mathcal{Q}^{\swarrow}$-RCP problem. Let $S \subseteq \mathbb{R}^2$ be a dataset of size n. Suppose $\Phi(S, \mathcal{Q}^{\swarrow}) = \{\phi_1, \ldots, \phi_m\}$ where $\phi_i = (a_i, b_i)$, and assume $\phi_1, \ldots, \phi_m$ are sorted in increasing order of their lengths. It was shown in [6] that $m = O(n)$. We construct a mapping $\Phi(S, \mathcal{Q}^{\swarrow}) \to \mathbb{R}^2$ as $\phi_i \mapsto w_i$ where $w_i = (\max\{a_i.x, b_i.x\}, \max\{a_i.y, b_i.y\})$, and observe that for a query range $Q \in \mathcal{Q}^{\swarrow}$, ϕ_i is contained in Q iff $w_i \in Q$. Let W_i be the northeast quadrant with vertex w_i. Then we further have $w_i \in Q$ iff $q \in W_i$ where q is the vertex of Q. As such, the closest-pair in $S \cap Q$ to be reported is ϕ_η for $\eta = \min\{i : q \in W_i\}$. We create a planar subdivision Γ, by successively overlaying $W_1, \ldots, W_m$ (see Figure 1). Note that the complexity of Γ is $O(m)$, since overlaying each quadrant creates at most two vertices of Γ. By the above observation, the answer for Q is ϕ_i iff q is in the cell $W_i \setminus \bigcup_{j=1}^{i-1} W_j$. Thus, we can use the optimal planar point-location data structures (e.g., [4, 8]) to solve the problem in $O(m)$ space with $O(\log m)$ query time. Since $m = O(n)$, we obtain a $\mathcal{Q}$-RCP data structure using $O(n)$ space with $O(\log n)$ query time in worst-case.

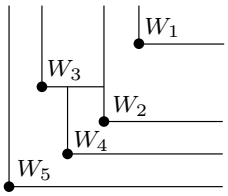


Figure 1 The subdivision induced by successively overlaying the quadrants.

Next, we analyze the average-case performance of the above data structure. In fact, it suffices to bound the expected number of the candidate pairs. Surprisingly, we have the following poly-logarithmic bound.

▶ **Lemma 2.** *For $S \propto R^n$ where R is an axis-parallel rectangle, $\mathbb{E}[|\Phi(S, \mathcal{Q})|] = O(\log^2 n)$.*

Proof Sketch. Assume $R = [0, 1] \times [0, \Delta]$ w.o.l.g. It suffices to show $\mathbb{E}[|\Phi(S, \mathcal{Q}^{\swarrow})|] = O(\log^2 n)$. Let $a_1, \ldots, a_n$ be the n random points in S, and $E_{i,j}$ be the event $(a_i, a_j) \in \Phi(S, \mathcal{Q}^{\swarrow})$. By linearity of expectation, one can see that $\mathbb{E}[|\Phi(S, \mathcal{Q}^{\swarrow})|] = O(n^2 \cdot \Pr[E_{1,2}])$. So it suffices to bound $\Pr[E_{1,2}]$. Define random variables $x_{\max} = \max\{a_1.x, a_2.x\}$, $y_{\max} = \max\{a_1.y, a_2.y\}$, and define $x_{\min}, y_{\min}$ similarly. The quadrant $Q = (-\infty, x_{\max}] \times (-\infty, y_{\max}]$ is the *minimal* quadrant containing a_1, a_2, and thus $(a_1, a_2) \in \Phi(S, \mathcal{Q}^{\swarrow})$ iff (a_1, a_2) is the closest-pair in $S \cap Q$. Define $\Lambda = \{i \geq 3 : a_i \in Q\}$, which is a random subset of $\{3, \ldots, n\}$.

We bound $\Pr[E_{1,2}]$ through several steps. First, we fix the values of $x_{\max}, y_{\max}, \Lambda$ and study the corresponding conditional probability of $E_{1,2}$. Fixing $\tilde{x} \in (0, 1]$, $\tilde{y} \in (0, \Delta]$, and nonempty $J \subseteq \{3, \ldots, n\}$, let $C_{\tilde{x},\tilde{y},J}$ be the event $(x_{\max} = \tilde{x}) \wedge (y_{\max} = \tilde{y}) \wedge (\Lambda = J)$. Consider $\Pr[E_{1,2} \mid C_{\tilde{x},\tilde{y},J}]$. Let $\delta_x = x_{\max} - x_{\min}$ and $\delta_y = y_{\max} - y_{\min}$. We observe that under the condition $C_{\tilde{x},\tilde{y},J}$, $E_{1,2}$ happens only if $(\delta_x \leq \kappa(S_J)) \wedge (\delta_y \leq \kappa(S_J))$, where $S_J = \{a_j : j \in J\}$. Furthermore, under $C_{\tilde{x},\tilde{y},J}$, the $|J|$ random points in S_J can be viewed as independently drawn from the uniform distribution on the rectangle $[0, \tilde{x}] \times [0, \tilde{y}]$. As such, Lemma 1 can be applied to understand the behavior of $\kappa(S_J)$. By properly applying Lemma 1 and doing some careful calculations, we deduce that under the condition $C_{\tilde{x},\tilde{y},J}$, $(\delta_x \leq \kappa(S_J)) \wedge (\delta_y \leq \kappa(S_J))$ happens with probablity $O(1/|J|^2)$. Thus, $\Pr[E_{1,2} \mid C_{\tilde{x},\tilde{y},J}] = O(1/|J|^2)$. This further implies $\Pr[E_{1,2} \mid |\Lambda| = m] = O(1/m^2)$ for all $m \in \{1, \ldots, n-2\}$. With this in hand, to bound $\Pr[E_{1,2}]$, it suffices to study $\Pr[|\Lambda| = m]$. To calculate $\Pr[|\Lambda| = m]$ is in fact a combinatorial

problem, because $|\Lambda|$ only depends on the orderings of the x-coordinates and y-coordinates of $a_1, \ldots, a_n$. Using combinatorial arguments, we obtain $\Pr[|\Lambda| = m] = O((m+1)\log n/n^2)$. Finally, applying the formula $\Pr[E_{1,2}] = \sum_{m=0}^{n-2} \Pr[|\Lambda| = m] \cdot \Pr[E_{1,2} \mid |\Lambda| = m]$ and the bounds achieved, we have $\Pr[E_{1,2}] = O(\log^2 n/n^2)$. As a result, $\mathbb{E}[|\Phi(S, \mathcal{Q}^{\swarrow})|] = O(\log^2 n)$ and $\mathbb{E}[|\Phi(S, \mathcal{Q})|] = O(\log^2 n)$. A complete proof can be found in [12]. ◀

Using the above lemma, we can immediately conclude that our data structure uses $O(\log^2 n)$ space in average-case. The average-case query time is in fact $O(\mathbb{E}[\log |\Phi(S, \mathcal{Q})|])$. Note that $\mathbb{E}[\log x] \leq \log \mathbb{E}[x]$ for a positive random variable x, thus $\mathbb{E}[\log |\Phi(S, \mathcal{Q})|] = O(\log \log n)$.

▶ **Theorem 3.** *There exists a $\mathcal{Q}$-RCP data structure $\mathcal{A}$ such that*

- *For any $S \subseteq \mathbb{R}^2$ of size n, $\mathsf{Space}(\mathcal{A}(S)) = O(n)$ and $\mathsf{Qtime}(\mathcal{A}(S)) = O(\log n)$.*
- *For a random $S \propto R^n$ where R is the unit square or more generally an arbitrary axis-parallel rectangle, $\mathbb{E}[\mathsf{Space}(\mathcal{A}(S))] = O(\log^2 n)$ and $\mathbb{E}[\mathsf{Qtime}(\mathcal{A}(S))] = O(\log \log n)$.*

3 Strip query

We consider the RCP problem for strip queries, i.e., the $\mathcal{P}$-RCP problem. In order to solve the $\mathcal{P}$-RCP problem, it suffices to consider the $\mathcal{P}^{\mathrm{v}}$-RCP problem. Let $S \subseteq \mathbb{R}^2$ be a dataset of size n. Suppose $\Phi(S, \mathcal{P}^{\mathrm{v}}) = \{\phi_1, \ldots, \phi_m\}$ where $\phi_i = (a_i, b_i)$, and assume $\phi_1, \ldots, \phi_m$ are sorted in increasing order of their lengths. It was shown in [10] that $m = O(n \log n)$. We construct a mapping $\Phi(S, \mathcal{P}^{\mathrm{v}}) \to \mathbb{R}^2$ as $\phi_i \mapsto w_i$ where $w_i = (\min\{a_i.x, b_i.x\}, \max\{a_i.x, b_i.x\})$, and observe that for a query range $P = [x_1, x_2] \times \mathbb{R} \in \mathcal{P}^{\mathrm{v}}$, ϕ_i is contained in P iff w_i is in the southeast quadrant $[x_1, \infty) \times (-\infty, x_2]$. Let W_i be the northwest quadrant with vertex w_i. Then we further have $w_i \in [x_1, \infty) \times (-\infty, x_2]$ iff $p \in W_i$ where $p = (x_1, x_2)$. As such, the closest-pair in $S \cap P$ is ϕ_η for $\eta = \min\{i : p \in W_i\}$. Thus, as in Section 2, we can successively overlay $W_1, \ldots, W_m$ to create a planar subdivision, and use point-location to solve the problem in $O(m)$ space and $O(\log m)$ query time. Since $m = O(n \log n)$ here, we obtain a $\mathcal{P}$-RCP data structure using $O(n \log n)$ space with $O(\log n)$ query time in worst-case.

Next, we analyze the average-case performance of our data structure. Again, it suffices to bound the expected number of the candidate pairs. For later use, we study here a more general case in which the queries are 3-sided rectangles.

▶ **Lemma 4.** *Let $S \propto \prod_{i=1}^{n} I_i$ where $I_1, \ldots, I_n$ are distinct vertical (resp., horizontal) aligned segments sorted from left to right (resp., from bottom to top). Suppose $a_i \in S$ is the point drawn on I_i. Then for $i, j \in \{1, \ldots, n\}$ with $i < j$ and $\mathcal{X} \in \{\mathcal{U}^{\downarrow}, \mathcal{U}^{\uparrow}\}$ (resp., $\mathcal{X} \in \{\mathcal{U}^{\leftarrow}, \mathcal{U}^{\rightarrow}\}$),*

$$\Pr[(a_i, a_j) \in \Phi(S, \mathcal{X})] = O\left(\frac{\log(j-i)}{(j-i)^2}\right).$$

From the above lemma, a direct calculation gives us the following corollary.

▶ **Corollary 5.** *For $S \propto R^n$ where R is an axis-parallel rectangle, $\mathbb{E}[|\Phi(S, \mathcal{U})|] = \Theta(n)$ and $\mathbb{E}[|\Phi(S, \mathcal{P})|] = \Theta(n)$.*

Using the above argument and our previous data structure, we conclude the following.

▶ **Theorem 6.** *There exists a $\mathcal{P}$-RCP data structure $\mathcal{B}$ such that*

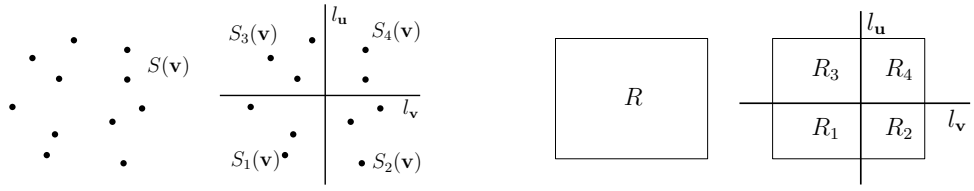
- *For any $S \subseteq \mathbb{R}^2$ of size n, $\mathsf{Space}(\mathcal{B}(S)) = O(n \log n)$ and $\mathsf{Qtime}(\mathcal{B}(S)) = O(\log n)$.*
- *For a random $S \propto R^n$ where R is the unit square or more generally an arbitrary axis-parallel rectangle, $\mathbb{E}[\mathsf{Space}(\mathcal{B}(S))] = O(n)$ and $\mathbb{E}[\mathsf{Qtime}(\mathcal{B}(S))] = O(\log n)$.*

4 Rectangle query

We consider the RCP problem for rectangle queries, i.e., the $\mathcal{R}$-RCP problem. Interestingly, our final solution for the $\mathcal{R}$-RCP problem is a combination of two simpler solutions, each of which partially achieves the desired bounds.

We first describe the common part of our two solutions. Let $S \subseteq \mathbb{R}^2$ be a dataset of size n. The common component of our two data structures is a standard 2D range tree built on S [3]. The main tree (or primary tree) $\mathcal{T}$ is a range tree built on the x-coordinates of the points in S. Each node $\mathbf{u} \in \mathcal{T}$ corresponds to a subset $S(\mathbf{u})$ of x-consecutive points in S, called the *canonical subset* of $\mathbf{u}$. At $\mathbf{u}$, there is an associated secondary tree $\mathcal{T}_\mathbf{u}$, which is a range tree built on the y-coordinates of the points in $S(\mathbf{u})$. With an abuse of notation, for each node $\mathbf{v} \in \mathcal{T}_\mathbf{u}$, we still use $S(\mathbf{v})$ to denote the canonical subset of $\mathbf{v}$, which is a subset of y-consecutive points in $S(\mathbf{u})$. As in [6], for each (non-leaf) primary node $\mathbf{u} \in \mathcal{T}$, we fix a vertical line $l_\mathbf{u}$ such that the points in the canonical subset of the left (resp., right) child of $\mathbf{u}$ are to the left (resp., right) of $l_\mathbf{u}$. Similarly, for each (non-leaf) secondary node $\mathbf{v}$, we fix a horizontal line $l_\mathbf{v}$ such that the points in the canonical subset of the left (resp., right) child of $\mathbf{v}$ are above (resp., below) $l_\mathbf{v}$. Let $\mathbf{v} \in \mathcal{T}_\mathbf{u}$ be a secondary node. Then at $\mathbf{v}$ we have two lines $l_\mathbf{v}$ and $l_\mathbf{u}$, which partition $\mathbb{R}^2$ into four quadrants. We denote by $S_1(\mathbf{v}), \ldots, S_4(\mathbf{v})$ the subsets of $S(\mathbf{v})$ contained in these quadrants; see Figure 2a for the correspondence. In order to solve the problem, we need to store some additional data structures at the nodes of the tree (called *sub-structures*). At each secondary node $\mathbf{v}$, we store four $\mathcal{Q}$-RCP data structures $\mathcal{A}(S_1(\mathbf{v})), \ldots, \mathcal{A}(S_4(\mathbf{v}))$ (Theorem 3).

Now let us explain what we can do by using this 2D range tree (with the sub-structures). Let $R = [x_1, x_2] \times [y_1, y_2] \in \mathcal{R}$ be a query rectangle. We first find in $\mathcal{T}$ the *splitting* node $\mathbf{u} \in \mathcal{T}$ corresponding to the range $[x_1, x_2]$, which is by definition the LCA of all the leaves whose corresponding points are in $[x_1, x_2] \times \mathbb{R}$. Then we find in $\mathcal{T}_\mathbf{u}$ the splitting node $\mathbf{v} \in \mathcal{T}_\mathbf{u}$ corresponding to the range $[y_1, y_2]$. If either of the splitting nodes does not exist or is a leaf node, then $|S \cap R| \leq 1$ and nothing should be reported. So assume $\mathbf{u}$ and $\mathbf{v}$ are non-leaf nodes. By the property of splitting node, we have $S \cap R = S(\mathbf{v}) \cap R$, and the lines $l_\mathbf{u}$ and $l_\mathbf{v}$ both intersect R. Thus, $l_\mathbf{u}$ and $l_\mathbf{v}$ decompose R into four smaller rectangles $R_1, \ldots, R_4$; see Figure 2b for the correspondence. By construction, we have $S(\mathbf{v}) \cap R_i = S_i(\mathbf{v}) \cap R_i$. In order to find the closest-pair in $S \cap R$, we first try to compute the closest-pair in $S \cap R_i$ for all $i \in \{1, \ldots, 4\}$. This can be done by querying the sub-structures stored at $\mathbf{v}$. Indeed, $S \cap R_i = S(\mathbf{v}) \cap R_i = S_i(\mathbf{v}) \cap R_i = S_i(\mathbf{v}) \cap Q_i$, where Q_i is the quadrant obtained by removing the two sides of R_i that coincide with $l_\mathbf{u}$ and $l_\mathbf{v}$. Therefore, we can query $\mathcal{A}(S_i(\mathbf{v}))$ with Q_i to find the closest-pair in $S \cap R_i$. Once the four closest-pairs are computed, we take the shortest one (i.e., the one of the smallest length) among them and denote it by ϕ.



(a) Illustrating the subsets $S_1(\mathbf{v}), \ldots, S_4(\mathbf{v})$. **(b)** Illustrating the rectangles $R_1, \ldots, R_4$.

Figure 2 Illustrating $S_1(\mathbf{v}), \ldots, S_4(\mathbf{v})$ and $R_1, \ldots, R_4$.

Clearly, ϕ is not necessarily the closest-pair in $S \cap R$ as the two points in the closest-pair may belong to different R_i's. However, as we will see, with ϕ in hand, finding the closest-pair in $S \cap R$ becomes easier. Suppose $l_{\mathbf{u}} : x = \alpha$ and $l_{\mathbf{v}} : y = \beta$, where $x_1 \leq \alpha \leq x_2$ and $y_1 \leq \beta \leq y_2$. Let δ be the length of ϕ. We define $P_\alpha = [\alpha - \delta, \alpha + \delta] \times \mathbb{R}$ (resp., $P_\beta = \mathbb{R} \times [\beta - \delta, \beta + \delta]$) and $R_\alpha = R \cap P_\alpha$ (resp., $R_\beta = R \cap P_\beta$); see Figure 3. We have the following key observation.

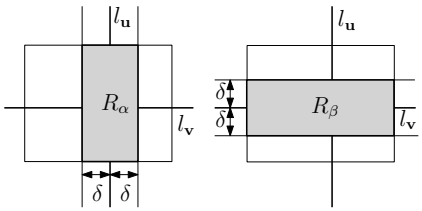


Figure 3 Illustrating the rectangles R_α and R_β.

▶ **Lemma 7.** *The closest-pair in $S \cap R$ is the shortest one among $\{\phi, \phi_\alpha, \phi_\beta\}$, where ϕ_α (resp., ϕ_β) is the closest-pair in $S \cap R_\alpha$ (resp., $S \cap R_\beta$).*

Due to the above lemma, it now suffices to compute ϕ_α and ϕ_β. Note that R_α and R_β are rectangles, so computing ϕ_α and ϕ_β still requires rectangle RCP queries. Fortunately, there are some additional properties which make it easy to search for the closest-pairs in $S \cap R_\alpha$ and $S \cap R_\beta$. For a set A of points in $\mathbb{R}^2$ and $a, b \in A$, we define the *x-gap* (resp., *y-gap*) between a and b in A as the number of the points in $A \backslash \{a, b\}$ whose x-coordinates (resp., y-coordinates) are in between $a.x$ and $b.x$ (resp., $a.y$ and $b.y$).

▶ **Lemma 8.** *There exists a constant integer k such that the y-gap (resp., x-gap) between the two points of ϕ_α (resp., ϕ_β) in $S \cap R_\alpha$ (resp., $S \cap R_\beta$) is at most k.*

We shall properly use the above lemma to help compute ϕ_α and ϕ_β. At this point, our two solutions diverge.

4.1 Preliminary: extreme point data structures

Before presenting our solutions, we introduce the so-called *top/bottom extreme point* (TBEP) and *left/right extreme point* (LREP) data structures. For a query space $\mathcal{X}$ and a constant integer k, an $(\mathcal{X}, k)$-TBEP (resp. $(\mathcal{X}, k)$-LREP) data structure stores a given set S of points in $\mathbb{R}^2$ and can report the k topmost/bottommost (resp., leftmost/rightmost) points in $S \cap X$ for a query range $X \in \mathcal{X}$.

▶ **Lemma 9.** *Let k be a constant integer. There exists a $(\mathcal{P}^{\mathrm{v}}, k)$-TBEP data structure $\mathcal{K}^{\mathrm{v}}$ such that for any $S \subseteq \mathbb{R}^2$ of size n, $\mathsf{Space}(\mathcal{K}^{\mathrm{v}}(S)) = O(n)$ and $\mathsf{Qtime}(\mathcal{K}^{\mathrm{v}}(S)) = O(\log n)$. Symmetrically, there also exists a $(\mathcal{P}^{\mathrm{h}}, k)$-LREP data structure $\mathcal{K}^{\mathrm{h}}$ satisfying the same bounds.*

▶ **Lemma 10.** *Let l be a vertical (resp., horizontal) line and k be a constant integer. There exists a $(\mathcal{P}_l, k)$-TBEP (resp., $(\mathcal{P}_l, k)$-LREP) data structure $\mathcal{K}_l$ such that for $S \propto \prod_{i=1}^{n} I_i$ where $I_1, \ldots, I_n$ are distinct vertical (resp., horizontal) aligned segments, $\mathbb{E}[\mathsf{Space}(\mathcal{K}_l(S))] = O(\log n)$ and $\mathbb{E}[\mathsf{Qtime}(\mathcal{K}_l(S))] = O(\log \log n)$.*

We remark here that the TBEP (resp., LREP) data structures are essentially top-k data structures when using the y-coordinates (resp., x-coordinates) as weights. Therefore, a 1D top-k data structure (see for example [7]) can be used to prove Lemma 9. For completeness, we also give a proof in the full version [12].

4.2 First solution

We now introduce our first solution, which achieves the desired worst-case bounds. Let k be the constant integer in Lemma 8. In our first solution, besides the 2D range tree presented

before, we build additionally two 1D range trees $\mathcal{T}'$ and $\mathcal{T}''$ on S, where $\mathcal{T}'$ (resp., $\mathcal{T}''$) is built on y-coordinates (resp., x-coordinates). For $\mathbf{u}' \in \mathcal{T}'$ (resp., $\mathbf{u}'' \in \mathcal{T}''$), we still use $S(\mathbf{u}')$ (resp., $S(\mathbf{u}'')$) to denote the canonical subset of $\mathbf{u}'$ (resp., $\mathbf{u}'' \in \mathcal{T}''$). At each node $\mathbf{u}' \in \mathcal{T}'$, we store a $\mathcal{P}$-RCP data structure $\mathcal{B}(S(\mathbf{u}'))$ (Theorem 6) and a $(\mathcal{P}^{\mathrm{v}}, k)$-TBEP data structure $\mathcal{K}^{\mathrm{v}}(S(\mathbf{u}'))$ (Lemma 9). Similarly, at each node $\mathbf{u}'' \in \mathcal{T}''$, we store a $\mathcal{P}$-RCP data structure $\mathcal{B}(S(\mathbf{u}''))$ (Theorem 6) and a $(\mathcal{P}^{\mathrm{h}}, k)$-LREP data structure $\mathcal{K}^{\mathrm{h}}(S(\mathbf{u}''))$ (Lemma 9).

We now explain how to compute ϕ_α and ϕ_β. Suppose $R_\alpha = [x_\alpha, x'_\alpha] \times [y_\alpha, y'_\alpha]$. Let $P_x = [x_\alpha, x'_\alpha] \times \mathbb{R}$ and $P_y = \mathbb{R} \times [y_\alpha, y'_\alpha]$. To compute ϕ_α, we first find in $\mathcal{T}'$ the $t = O(\log n)$ canonical nodes $\mathbf{u}'_1, \ldots, \mathbf{u}'_t \in \mathcal{T}'$ corresponding to the range $[y_\alpha, y'_\alpha]$. Then $\bigcup_{i=1}^{t} S(\mathbf{u}'_i) = S \cap P_y$, and each $S(\mathbf{u}'_i)$ is a set of y-consecutive points in $S \cap P_y$. Furthermore, $S \cap R_\alpha = \bigcup_{i=1}^{t} S(\mathbf{u}'_i) \cap P_x$. We query the sub-structures $\mathcal{B}(S(\mathbf{u}'_1)), \ldots, \mathcal{B}(S(\mathbf{u}'_t))$ with P_x to find the closest-pairs $\phi_1, \ldots, \phi_t$ in $S(\mathbf{v}_1) \cap P_x, \ldots, S(\mathbf{v}_t) \cap P_x$, respectively. We also query $\mathcal{K}^{\mathrm{v}}(S(\mathbf{u}'_1)), \ldots, \mathcal{K}^{\mathrm{v}}(S(\mathbf{u}'_t))$ with P_x to obtain the k topmost and bottommost points in $S(\mathbf{u}'_1) \cap P, \ldots, S(\mathbf{u}'_t) \cap P$, respectively; we denote by K the set of the $2kt$ reported points. Then we find the closest-pair ϕ_K in K using the standard divide-and-conquer algorithm. We claim that ϕ_α is the shortest one among $\{\phi_1, \ldots, \phi_t, \phi_K\}$. Suppose $\phi_\alpha = (a, b)$. If the two points of ϕ_α are both contained in some $S(\mathbf{u}'_i)$, then clearly $\phi_\alpha = \phi_i$. Otherwise, by Lemma 8 and the choice of k, the two points of ϕ_α must belong to K and hence $\phi_\alpha = \phi_K$. It follows that $\phi_\alpha \in \{\phi_1, \ldots, \phi_t, \phi_K\}$. Furthermore, because the pairs $\phi_1, \ldots, \phi_t, \phi_K$ are all contained in R_α, ϕ_α must be the shortest one among $\{\phi_1, \ldots, \phi_t, \phi_K\}$. Therefore, with $\phi_1, \ldots, \phi_t, \phi_K$ in hand, ϕ_α can be easily computed. The pair ϕ_β is computed symmetrically using $\mathcal{T}''$. Finally, taking the shortest one among $\{\phi, \phi_\alpha, \phi_\beta\}$, the query R can be answered.

The 2D range tree together with the two 1D range trees $\mathcal{T}'$ and $\mathcal{T}''$ forms an $\mathcal{R}$-RCP data structure, which is our first solution. A straightforward analysis gives us the worst-case space cost and query time of this data structure.

▶ **Theorem 11.** *There exists an $\mathcal{R}$-RCP data structure $\mathcal{D}_1$ such that for any $S \subseteq \mathbb{R}^2$ of size n, $\mathsf{Space}(\mathcal{D}_1(S)) = O(n \log^2 n)$ and $\mathsf{Qtime}(\mathcal{D}_1(S)) = O(\log^2 n)$.*

Our first solution itself already achieves the desired worst-case bounds, which simultaneously improves the results given in [6] and [10].

4.3 Second solution

We now introduce our second solution, which has the desired average-case space cost and an $O(\log n)$ query time (even in worst-case). In our second solution, we only use the 2D range tree presented before, but we need some additional sub-structures stored at each secondary node. Let k be the constant integer in Lemma 8. Define $S_{\blacktriangle}(\mathbf{v}) = S_3(\mathbf{v}) \cup S_4(\mathbf{v})$ (resp., $S_{\blacktriangledown}(\mathbf{v}) = S_1(\mathbf{v}) \cup S_2(\mathbf{v})$) as the subset of $S(\mathbf{v})$ consisting of the points above (resp., below) $l_{\mathbf{v}}$. Similarly, define $S_{\blacktriangleleft}(\mathbf{v})$ and $S_{\blacktriangleright}(\mathbf{v})$ as the subsets to the left and right of $l_{\mathbf{u}}$, respectively. Let $\mathbf{v} \in \mathcal{T}_{\mathbf{u}}$ be a secondary node. Besides $\mathcal{A}(S_1(\mathbf{v})), \ldots, \mathcal{A}(S_4(\mathbf{v}))$, we store at $\mathbf{v}$ two $(\mathcal{P}_{l_{\mathbf{u}}}, k)$-TBEP data structures $\mathcal{K}_{l_{\mathbf{u}}}(S_{\blacktriangle}(\mathbf{v})), \mathcal{K}_{l_{\mathbf{u}}}(S_{\blacktriangledown}(\mathbf{v}))$ (Lemma 10) and two $(\mathcal{P}_{l_{\mathbf{v}}}, k)$-LREP data structures $\mathcal{K}_{l_{\mathbf{v}}}(S_{\blacktriangleleft}(\mathbf{v})), \mathcal{K}_{l_{\mathbf{v}}}(S_{\blacktriangleright}(\mathbf{v}))$ (Lemma 10). Furthermore, we need a new kind of sub-structures called *range shortest-segment* (RSS) data structures. For a query space $\mathcal{X}$, an $\mathcal{X}$-RSS data structure stores a given set of segments in $\mathbb{R}^2$ and can report the shortest segment contained in a query range $X \in \mathcal{X}$. We have the following observation.

▶ **Lemma 12.** *There exists a $\mathcal{U}$-RSS data structure $\mathcal{C}$ such that for any set G of m segments in $\mathbb{R}^2$, $\mathsf{Space}(\mathcal{C}(G)) = O(m^2)$ and $\mathsf{Qtime}(\mathcal{C}(G)) = O(\log m)$.*

Define $\Phi_{\blacktriangle}(\mathbf{v}) = \Phi_{l_{\mathbf{u}}}(S_{\blacktriangle}(\mathbf{v}), \mathcal{U}^{\downarrow})$, $\Phi_{\blacktriangledown}(\mathbf{v}) = \Phi_{l_{\mathbf{u}}}(S_{\blacktriangledown}(\mathbf{v}), \mathcal{U}^{\uparrow})$, $\Phi_{\blacktriangleleft}(\mathbf{v}) = \Phi_{l_{\mathbf{v}}}(S_{\blacktriangleleft}(\mathbf{v}), \mathcal{U}^{\rightarrow})$, $\Phi_{\blacktriangleright}(\mathbf{v}) = \Phi_{l_{\mathbf{v}}}(S_{\blacktriangleright}(\mathbf{v}), \mathcal{U}^{\leftarrow})$. We can view $\Phi_{\blacktriangle}(\mathbf{v}), \Phi_{\blacktriangledown}(\mathbf{v}), \Phi_{\blacktriangleleft}(\mathbf{v}), \Phi_{\blacktriangleright}(\mathbf{v})$ as four sets of segments by identifying each point-pair (a, b) as a segment $[a, b]$. Then we store at $\mathbf{v}$ four $\mathcal{U}$-RSS data structures $\mathcal{C}(\Phi_{\blacktriangle}(\mathbf{v})), \mathcal{C}(\Phi_{\blacktriangledown}(\mathbf{v})), \mathcal{C}(\Phi_{\blacktriangleleft}(\mathbf{v})), \mathcal{C}(\Phi_{\blacktriangle}(\mathbf{v}))$ (Lemma 12).

We now explain how to compute ϕ_α and ϕ_β. Let us consider ϕ_α. Recall that ϕ_α is the closest-pair in $S \cap R_\alpha$, i.e., in $S(\mathbf{v}) \cap R_\alpha$. Let P be the $l_{\mathbf{u}}$-anchored strip obtained by removing the top/bottom bounding line of R_α. If the two points of ϕ_α are on opposite sides of $l_{\mathbf{v}}$, then by Lemma 8 its two points must be among the k bottommost points in $S_{\blacktriangle}(\mathbf{v}) \cap P$ and the k topmost points in $S_{\blacktriangledown}(\mathbf{v}) \cap P$ respectively. Using $\mathcal{K}_{l_{\mathbf{u}}}(S_{\blacktriangle}(\mathbf{v}))$ and $\mathcal{K}_{l_{\mathbf{u}}}(S_{\blacktriangledown}(\mathbf{v}))$, we report these $2k$ points, and compute the closest-pair among them by brute-force. If the two points of ϕ_α are on the same side of $l_{\mathbf{v}}$, then they are both contained in either $S_{\blacktriangle}(\mathbf{v})$ or $S_{\blacktriangledown}(\mathbf{v})$. So it suffices to compute the closest-pairs in $S_{\blacktriangle}(\mathbf{v}) \cap R_\alpha$ and $S_{\blacktriangledown}(\mathbf{v}) \cap R_\alpha$. Without loss of generality, we only need to consider the closest-pair in $S_{\blacktriangle}(\mathbf{v}) \cap R_\alpha$. We denote by U the 3-sided rectangle obtained by removing the bottom boundary of R_α, and by Q_1 (resp., Q_2) the quadrant obtained by removing the right (resp., left) boundary of U. We query $\mathcal{A}(S_1(\mathbf{v}))$ with Q_1, $\mathcal{A}(S_2(\mathbf{v}))$ with Q_2, and $\mathcal{C}(\Phi_{\blacktriangle}(\mathbf{v}))$ with U. Clearly, the shortest one among the three answers is the closest-pair in $S_{\blacktriangle}(\mathbf{v}) \cap R_\alpha$. Indeed, the three answers are all point-pairs in $S_{\blacktriangle}(\mathbf{v}) \cap R_\alpha$. If the two points of the closest-pair in $S_{\blacktriangle}(\mathbf{v}) \cap R_\alpha$ are both to the left (resp., right) of $l_{\mathbf{u}}$, $\mathcal{A}(S_1(\mathbf{v}))$ (resp., $\mathcal{A}(S_2(\mathbf{v}))$) reports it; otherwise, the closest-pair crosses $l_{\mathbf{u}}$, and $\mathcal{C}(\Phi_{\blacktriangle}(\mathbf{v}))$ reports it. Now we see how to compute ϕ_α, and ϕ_β can be computed symmetrically. Finally, taking the shortest one among $\{\phi, \phi_\alpha, \phi_\beta\}$, the query R can be answered.

A straightforward analysis shows that the overall query time is $O(\log n)$ even in worst-case. The worst-case space cost is not near-linear, as the $\mathcal{U}$-RSS data structure $\mathcal{C}$ may occupy quadratic space by Lemma 12. However, we can show that the average-case space cost is in fact $O(n \log n)$. The crucial thing is to bound the average-case space of the sub-structures stored at the secondary nodes. The intuition for bounding the average-case space of the $\mathcal{Q}$-RCP and TBEP/LREP sub-structures comes directly from the average-case performance of our $\mathcal{Q}$-RCP data structure (Theorem 3) and TBEP/LREP data structure (Lemma 10). However, to bound the average-case space of the $\mathcal{U}$-RSS sub-structures is more difficult. By our construction, the segments stored in these sub-structures are 3-sided candidate pairs that cross a line. As such, we have to study the expected number of such candidate pairs in a random dataset. To this end, we recall Lemma 4. Let l be a vertical line, and $S \propto \prod_{i=1}^{n} I_i$ be a random dataset drawn from vertical aligned segments $I_1, \ldots, I_n$ as in Lemma 4. Suppose we build a $\mathcal{U}$-RSS data structure $\mathcal{C}(\Phi)$ on $\Phi = \Phi_l(S, \mathcal{U}^{\downarrow})$. Using Lemma 4, a direct calculation gives us $\mathbb{E}[|\Phi_l(S, \mathcal{U}^{\downarrow})|] = O(\log^2 n)$. Unfortunately, this is not sufficient for bounding the average-case space of $\mathcal{C}(\Phi)$, because $\mathbb{E}[\mathsf{Space}(\mathcal{C}(\Phi))] = O(\mathbb{E}[|\Phi_l(S, \mathcal{U}^{\downarrow})|^2])$ and in general $\mathbb{E}[|\Phi_l(S, \mathcal{U}^{\downarrow})|^2] \neq \mathbb{E}^2[|\Phi_l(S, \mathcal{U}^{\downarrow})|]$. Therefore, we need a bound for $\mathbb{E}[|\Phi_l(S, \mathcal{U}^{\downarrow})|^2]$, which can also be obtained using Lemma 4, but requires more work.

▶ **Lemma 13.** *Let l be a vertical (resp., horizontal) line and $S \propto \prod_{i=1}^{n} I_i$ where $I_1, \ldots, I_n$ are distinct vertical (resp., horizontal) aligned segments. Then for $\mathcal{X} \in \{\mathcal{U}^{\downarrow}, \mathcal{U}^{\uparrow}\}$ (resp., $\mathcal{X} \in \{\mathcal{U}^{\leftarrow}, \mathcal{U}^{\rightarrow}\}$) , $\mathbb{E}[|\Phi_l(S, \mathcal{X})|] = O(\log^2 n)$ and $\mathbb{E}[|\Phi_l(S, \mathcal{X})|^2] = O(\log^4 n)$.*

Now we are ready to prove the bounds of our second solution.

▶ **Theorem 14.** *There exists an $\mathcal{R}$-RCP data structure $\mathcal{D}_2$ such that*

- *For any $S \subseteq \mathbb{R}^2$ of size n, $\mathsf{Qtime}(\mathcal{D}_2(S)) = O(\log n)$.*
- *For a random $S \propto R^n$ where R is the unit square or more generally an arbitrary axis-parallel rectangle, $\mathbb{E}[\mathsf{Space}(\mathcal{D}_2(S))] = O(n \log n)$.*

Proof sketch. The query time can be shown via a direct analysis. To bound the average-case space cost, let $S \propto R^n$. Since a 2D range tree built on a dataset of n points has a fixed

tree structure independent of the dataset (while depending on the number n), $\mathcal{D}_2(S)$ can be viewed as a fixed 2D range tree with random sub-structures. Let $\mathbf{v} \in \mathcal{T}_\mathbf{u}$ be a secondary node. We want to bound the expected space cost of the sub-structures stored at $\mathbf{v}$. To this end, we take advantage of Theorem 3, Lemma 10, and Lemma 13. However, before this, there is a crucial issue to be handled. We notice that Theorem 3, Lemma 10, and Lemma 13 assume the random dataset is independently and uniformly generated from either an axis-parallel rectangle or a set of aligned segments. Unfortunately, the underlying datasets of the sub-structures stored at $\mathbf{v}$, which are $S_1(\mathbf{v}), \ldots, S_4(\mathbf{v})$ and $S_{\blacktriangle}(\mathbf{v}), S_{\blacktriangledown}(\mathbf{v}), S_{\blacktriangleleft}(\mathbf{v}), S_{\blacktriangleright}(\mathbf{v})$, are neither (independently and uniformly) generated from a rectangle nor generated from aligned segments. To handle this issue is the technical part of this proof, and we only give some intuition here (the details can be found in the complete proof). The key idea is to fix a configuration of the points outside the random dataset under consideration, which makes the random dataset distributed independently and uniformly on a certain rectangle. For instance, if we fix a configuration of $S \backslash S_1(\mathbf{v})$, then $S_1(\mathbf{v})$ can be viewed as independently and uniformly generated from a rectangle and thus Theorem 3 applies to bound the expected space cost of $\mathcal{A}(S_1(\mathbf{v}))$. In this way, we show that the expected space cost of the sub-structures stored at $\mathbf{v}$ is poly-logarithmic in $|S(\mathbf{v})|$. It follows immediately that $\mathbb{E}[\mathsf{Space}(\mathcal{D}_2(S))] = O(n \log n)$. A complete proof can be found in [12]. ◀

4.4 Combining the two solutions

We now combine the two data structures $\mathcal{D}_1$ (Theorem 11) and $\mathcal{D}_2$ (Theorem 14) to obtain a *single* data structure $\mathcal{D}$ that achieves the desired worst-case and average-case bounds simultaneously. For a dataset $S \subseteq \mathbb{R}^2$ of size n, if $\mathsf{Space}(\mathcal{D}_2(S)) \geq n \log^2 n$, we set $\mathcal{D}(S) = \mathcal{D}_1(S)$, otherwise we set $\mathcal{D}(S) = \mathcal{D}_2(S)$. The worst-case bounds of $\mathcal{D}$ follows directly, while the average-case bounds follows from an analysis using Markov's inequality.

▶ **Theorem 15.** *There exists an $\mathcal{R}$-RCP data structure $\mathcal{D}$ such that*

- *For any $S \subseteq \mathbb{R}^2$ of size n, $\mathsf{Space}(\mathcal{D}(S)) = O(n \log^2 n)$ and $\mathsf{Qtime}(\mathcal{D}(S)) = O(\log^2 n)$.*
- *For a random $S \propto R^n$ where R is the unit square or more generally an arbitrary axis-parallel rectangle, $\mathbb{E}[\mathsf{Space}(\mathcal{D}(S))] = O(n \log n)$ and $\mathbb{E}[\mathsf{Qtime}(\mathcal{D}(S))] = O(\log n)$.*

5 Halfplane query

We consider the RCP problem for halfplane queries, i.e., the $\mathcal{H}$-RCP problem. In order to solve the $\mathcal{H}$-RCP problem, it suffices to consider the $\mathcal{H}^{\uparrow}$-RCP problem. Let $S \subseteq \mathbb{R}^2$ be the dataset of size n.

We shall apply the standard duality technique [3]. A non-vertical line $l : y = ux + v$ in $\mathbb{R}^2$ is dual to the point $l^* = (u, -v)$ and a point $p = (s, t) \in \mathbb{R}^2$ is dual to the line $p^* : y = sx - t$. A basic property of duality is that $p \in l^{\uparrow}$ (resp., $p \in l^{\downarrow}$) iff $l^* \in (p^*)^{\uparrow}$ (resp., $l^* \in (p^*)^{\downarrow}$). To make the exposition cleaner, we distinguish between *primal space* and *dual space*, which are two copies of $\mathbb{R}^2$. The dataset S and query ranges are assumed to lie in the primal space, while their dual objects are assumed to lie in the dual space. Duality allows us to transform the $\mathcal{H}^{\uparrow}$-RCP problem into a point location problem as follows. Let $H = l^{\uparrow} \in \mathcal{H}^{\uparrow}$ be a query range. The line l bounding H is dual to the point l^* in the dual space; for convenience, we

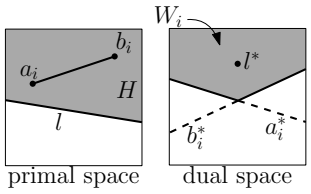


Figure 4 Illustrating the upward-open wedge W_i.

also call l^* the dual point of H. If we decompose the dual space into "cells" such that the query ranges whose dual points lie in the same cell have the same answer, then point location techniques can be applied to solve the problem directly. Note that this decomposition must be a polygonal subdivision Γ of $\mathbb{R}^2$, which consists of vertices, straight-line edges, and polygonal faces (i.e., cells). This is because the cell-boundaries must be defined by the dual lines of the points in S. In order to analyze the space cost and query time, we need to study the complexity $|\Gamma|$ of Γ. An $O(n^2)$ trivial upper bound for $|\Gamma|$ follows from the fact that the subdivision formed by the n dual lines of the points in S has an $O(n^2)$ complexity. In what follows, we shall show $|\Gamma| = O(n)$ by using the additional properties of the problem, which is a key ingredient of our result in this section.

Suppose $\Phi(S, \mathcal{H}^{\uparrow}) = \{\phi_1, \ldots, \phi_m\}$ where $\phi_i = (a_i, b_i)$ and $\phi_1, \ldots, \phi_m$ are sorted in increasing order of their lengths. It was shown in [1] that $m = O(n)$, and the candidate pairs do not cross each other, i.e., the segments $[a_i, b_i]$ and $[a_j, b_j]$ do not cross for any $i \neq j$. The non-crossing property of the candidate pairs is important and will be used later for proving Lemma 16. With this in hand, we now consider the subdivision Γ. Let $H = l^{\uparrow} \in \mathcal{H}^{\uparrow}$ be a query range. By the property of duality, ϕ_i is contained in H iff $l^* \in (a_i^*)^{\uparrow}$ and $l^* \in (b_i^*)^{\uparrow}$, i.e., l^* is in the upward-open *wedge* W_i generated by the lines a_i^* and b_i^* (in the dual space); see Figure 4.

As such, the closest-pair in $S \cap H$ to be reported is ϕ_η for $\eta = \min\{i : l^* \in W_i\}$. Therefore, Γ can be constructed by successively overlaying the wedges $W_1, \ldots, W_m$ (similarly to what we see in Section 2). Formally, we begin with a trivial subdivision Γ_0 of $\mathbb{R}^2$, which consists of only one face, the entire plane. Suppose Γ_{i-1} is constructed, which has an *outer face* F_{i-1} equal to the complement of $\bigcup_{j=1}^{i-1} W_j$ in $\mathbb{R}^2$. Now we construct a new subdivision Γ_i by "inserting" W_i to Γ_{i-1}. Specifically, Γ_i is obtained from Γ_{i-1} by decomposing the outer face F_{i-1} via the wedge W_i; that is, we decompose F_{i-1} into several smaller faces: one is $F_{i-1} \backslash W_i$ and the others are the connected components of $F_{i-1} \cap W_i$. Note that $F_{i-1} \backslash W_i$ is the complement of $\bigcup_{j=1}^{i} W_j$, which is connected (as one can easily verify) and becomes the outer face F_i of Γ_i. In this way, we construct $\Gamma_1, \ldots, \Gamma_m$ in order, and it is clear that $\Gamma_m = \Gamma$. The linear upper bound for $|\Gamma|$ follows from the following technical result.

▶ **Lemma 16.** *$|\Gamma_i| - |\Gamma_{i-1}| = O(1)$ for $i \in \{1, \ldots, m\}$. In particular, $|\Gamma| = O(m)$.*

Proof sketch. We denote by ∂W_i the boundary of the wedge W_i, which consists of two rays (emanating from a point) contained in a_i^* and b_i^* respectively. We observe that, to prove $|\Gamma_i| - |\Gamma_{i-1}| = O(1)$, it suffices to show the number of the connected components of $\partial W_i \cap F_{i-1}$ is constant. This can be further reduced to considering one ray of ∂W_i (say the ray r contained in a_i^*). We notice that $r \cap F_{i-1} = r \backslash \bigcup_{j=1}^{i-1} (r \cap W_j)$, and each $r \cap W_j$ is a connected portion of r. Let l_i be the line through a_i, b_i and l_i' be the line through a_i that is perpendicular to l_i, then l_i^* is the initial point of r and $(l_i')^*$ is a point on a_i^*. The crucial observation here is that each $r \cap W_j$ for $j \in \{1, \ldots, i-1\}$ satisfies at least one of the four conditions: **(i)** $r \cap W_j$ is empty; **(ii)** $r \cap W_j$ contains the initial point of r; **(iii)** $r \cap W_j$ contains the infinite end of r; **(iv)** $r \cap W_j$ contains the point $(l_i')^*$. The proof of this observation requires us to carefully analyze various cases using the properties of duality and the problem itself. Basically, we consider three cases: (1) $a_j, b_j \in l_i^{\uparrow}$; (2) $a_j, b_j \in l_i^{\downarrow}$; (3) one of a_j, b_j is strictly above l_i while the other is strictly below l_i. The first two cases are not very difficult, and are analyzed using duality and some geometry. The last case is the most subtle one. To handle it requires a careful use of the facts that ϕ_i, ϕ_j do not cross (i.e., the segments $[a_i, b_i]$ and $[a_j, b_j]$ do not cross) and ϕ_j is shorter than ϕ_i (because $j < i$ and $\phi_1, \ldots, \phi_m$ are sorted in increasing order of their lengths), as well as some properties of duality. We omit the detailed analysis in this sketch. Once the observation is proved, it

follows readily that $\bigcup_{j=1}^{i-1}(r \cap W_j)$ has at most three connected components and $r \cap F_{i-1}$ has at most two. As such, $|\Gamma_i| - |\Gamma_{i-1}| = O(1)$. A complete proof can be found in [12]. ◀

With the above result in hand, we can build an optimal point-location data structure for Γ using $O(m)$ space with $O(\log m)$ query time to solve the RCP problem. Since $m = O(n)$, we obtain an $\mathcal{H}$-RCP data structure using $O(n)$ space and $O(\log n)$ query time in worst-case.

Next, we analyze the average-case bounds of the above data structure. In fact, it suffices to bound the expected number of the candidate pairs. Similarly to the quadrant case, we can prove a poly-logarithmic bound.

▶ **Lemma 17.** *For $S \propto R^n$ where R is an axis-parallel rectangle, $\mathbb{E}[|\Phi(S, \mathcal{H})|] = O(\log^2 n)$.*

Now we are able to conclude the following.

▶ **Theorem 18.** *There exists an $\mathcal{H}$-RCP data structure $\mathcal{E}$ such that*

- *For any $S \subseteq \mathbb{R}^2$ of size n, $\mathsf{Space}(\mathcal{E}(S)) = O(n)$ and $\mathsf{Qtime}(\mathcal{E}(S)) = O(\log n)$.*
- *For a random $S \propto R^n$ where R is the unit square or more generally an arbitrary axis-parallel rectangle, $\mathbb{E}[\mathsf{Space}(\mathcal{E}(S))] = O(\log^2 n)$ and $\mathbb{E}[\mathsf{Qtime}(\mathcal{E}(S))] = O(\log \log n)$.*

Our data structure can be built in worst-case $O(n \log^2 n)$ time. We only give the high-level idea here, and the details can be found in [12]. We first observe that if the candidate pairs $\phi_1, \ldots, \phi_m$ are already given, then the subdivision Γ can be constructed in $O(m \log m)$ time. The idea is to begin with Γ_0 and iteratively construct Γ_i from Γ_{i-1} by "inserting" the wedge W_i dual to ϕ_i. Each Γ_i can be constructed in *amortized* $O(\log m)$ time (from Γ_{i-1}) by using a (balanced) BST to maintain the outer face and properly exploiting the behavior of W_i observed in Lemma 16. It follows that Γ can be constructed in $O(m \log m)$ time. Now consider the general case in which $\phi_1, \ldots, \phi_m$ are not given. We use an approach in [1] to compute in $O(n \log^2 n)$ time a set Ψ of $O(n \log n)$ point-pairs in S such that $\Phi(S, \mathcal{H}^{\uparrow}) \subseteq \Psi$. By considering the pairs in Ψ in increasing order of their lengths, we can efficiently verify whether each pair is a candidate pair or not, and update the subdivision (using the method above) whenever a candidate pair is recognized. The overall process takes $O(n \log^2 n)$ time.

6 Conclusion and future work

We revisited the range closest-pair (RCP) problem, which aims to preprocess a set S of points in $\mathbb{R}^2$ into a data structure such that for any query range X, the closest-pair in $S \cap X$ can be reported efficiently. We proposed new RCP data structures for various query types (including quadrants, strips, rectangles, and halfplanes). Both worst-case and average-case analyses were applied, resulting in new bounds for the RCP problem (see Table 1).

We now list some open questions for future study. First, as mentioned in Section 1.1, the preprocessing for our orthogonal RCP data structures remains open. It is not clear how to build these data structures in sub-quadratic time. Besides, the RCP problem for other query types is also open. One important example is the disk query, which is usually much harder than the rectangle query and halfplane query in traditional range search. For an easier version, we can focus on the case where the query disks have a fixed radius, or equivalently, the query ranges are *translates* of a fixed disk. Along this direction, one can also consider translation queries of some shape other than a disk. For instance, if the query ranges are translates of a fixed rectangle, can we have more efficient data structures than our rectangle RCP data structure in Section 4? Finally, the RCP problem in higher dimensions is quite open. To our best knowledge, the only known result for this is a simple data structure given in [6] constructed by explicitly storing all the candidate pairs, which only has guaranteed average-case performance.

References

1. Mohammad Ali Abam, Paz Carmi, Mohammad Farshi, and Michiel Smid. On the power of the semi-separated pair decomposition. In *Workshop on Algorithms and Data Structures*, pages 1–12. Springer, 2009.
2. Pankaj K Agarwal, Jeff Erickson, et al. Geometric range searching and its relatives. *Contemporary Mathematics*, 223:1–56, 1999.
3. M. de Berg, M. van Kreveld, M. Overmars, and O. C. Schwarzkopf. Computational geometry. In *Computational geometry*, pages 1–17. Springer, 2000.
4. Herbert Edelsbrunner, Leonidas J Guibas, and Jorge Stolfi. Optimal point location in a monotone subdivision. *SIAM Journal on Computing*, 15(2):317–340, 1986.
5. Prosenjit Gupta. Range-aggregate query problems involving geometric aggregation operations. *Nordic journal of Computing*, 13(4):294–308, 2006.
6. Prosenjit Gupta, Ravi Janardan, Yokesh Kumar, and Michiel Smid. Data structures for range-aggregate extent queries. *Computational Geometry: Theory and Applications*, 2(47):329–347, 2014.
7. Saladi Rahul and Yufei Tao. On top-k range reporting in 2D space. In *Proceedings of the 34th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 265–275. ACM, 2015.
8. Neil Sarnak and Robert E Tarjan. Planar point location using persistent search trees. *Communications of the ACM*, 29(7):669–679, 1986.
9. Jing Shan, Donghui Zhang, and Betty Salzberg. On spatial-range closest-pair query. In *International Symposium on Spatial and Temporal Databases*, pages 252–269. Springer, 2003.
10. R Sharathkumar and Prosenjit Gupta. Range-aggregate proximity queries. *IIIT Hyderabad, Telangana*, 500032, 2007.
11. Michiel Smid. *Closest point problems in computational geometry*. Citeseer, 1995.
12. Jie Xue, Yuan Li, Saladi Rahul, and Ravi Janardan. New bounds for range closest-pair problems. *arXiv preprint arXiv:1712.09749*, 2017.

Coordinated Motion Planning: The Video

Aaron T. Becker[1]
Department of Electrical and Computer Engineering, University of Houston
Houston, TX 77204-4005 USA
atbecker@uh.edu
https://orcid.org/0000-0001-7614-6282

Sándor P. Fekete
Department of Computer Science, TU Braunschweig
Mühlenpfordtstr. 23, 38106 Braunschweig, Germany
s.fekete@tu-bs.de
https://orcid.org/0000-0002-9062-4241

Phillip Keldenich[2]
Department of Computer Science, TU Braunschweig
Mühlenpfordtstr. 23, 38106 Braunschweig, Germany
p.keldenich@tu-bs.de
https://orcid.org/0000-0002-6677-5090

Matthias Konitzny
Department of Computer Science, TU Braunschweig
Mühlenpfordtstr. 23, 38106 Braunschweig, Germany
m.konitzny@tu-bs.de

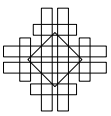
Lillian Lin[3]
Department of Electrical and Computer Engineering, University of Houston
Houston, TX 77204-4005 USA
atbecker@uh.edu
https://orcid.org/0000-0002-2834-9234

Christian Scheffer
Department of Computer Science, TU Braunschweig
Mühlenpfordtstr. 23, 38106 Braunschweig, Germany
c.scheffer@tu-bs.de
https://orcid.org/0000-0002-3471-2706

Abstract

We motivate, visualize and demonstrate recent work for minimizing the total execution time of a coordinated, *parallel* motion plan for a swarm of N robots in the absence of obstacles. Under relatively mild assumptions on the separability of robots, the algorithm achieves *constant stretch*: If all robots want to move at most d units from their respective starting positions, then the total duration of the overall schedule (and hence the distance traveled by each robot) is $\mathcal{O}(d)$ steps; this implies constant-factor approximation for the optimization problem. Also mentioned is an NP-hardness result for finding an optimal schedule, even in the case in which robot positions are restricted to a regular grid. On the other hand, we show that for densely packed disks that cannot be well separated, a stretch factor $\Omega(N^{1/4})$ is required in the worst case; we establish an achievable stretch factor of $O(N^{1/2})$ even in this case. We also sketch geometric difficulties of computing optimal trajectories, even for just two unit disks.

[1] Supported by the National Science Foundation under Grant No. IIS-1553063.
[2] Supported by the German Research Foundation under Grant No. FE 407/17-2.
[3] Supported by the National Science Foundation under Grant No. IIS-1553063.



34th International Symposium on Computational Geometry (SoCG 2018).
Editors: Csaba Tóth and Bettina Speckmann; Article No. 74; pp. 74:1–74:6

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

2012 ACM Subject Classification Theory of computation → Computational geometry, Theory of computation → Problems, reductions and completeness, Computer systems organization → Robotic control

Keywords and phrases Motion planning, robot swarms, complexity, stretch, approximation

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.74

Category Multimedia Exposition

Related Version The paper described in this contribution is [1], `http://dx.doi.org/10.4230/LIPIcs.SoCG.2018.29`; the full version can be found at `https://arxiv.org/abs/1801.01689`. The video associated with this abstract can be reached via `http://computational-geometry.org/SoCG-videos/socg18video/videos/74`.

Acknowledgements We thank our coauthors of paper [1], Erik Demaine, Henk Meijer for many helpful and motivating discussions. We also thank anonymous reviewers for helpful suggestions.

1 Introduction

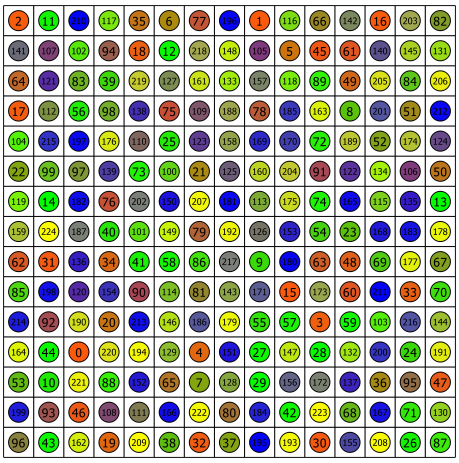
From the early days of computational geometry, robot motion planning has received a large amount of algorithmic attention. In the groundbreaking work by Schwartz and Sharir [2] from the 1980s, one of the challenges was coordinating the motion of *several* disk-shaped objects among obstacles. Their algorithms run in time polynomial in the complexity of the obstacles, but exponential in the number of disks, illustrating the significant challenge of coordinating many individual robots.

More recently, a growing spectrum of applications has increased the importance of multi-robot motion planning. However, previous work has largely focused on *sequential* schedules, in which one robot moves at a time, with objectives such as the number of moves; this differs from the practically important task of minimizing the overall makespan (i.e., the total time until completion) of a coordinated *parallel* motion schedule in which many robots are allowed to move simultaneously.

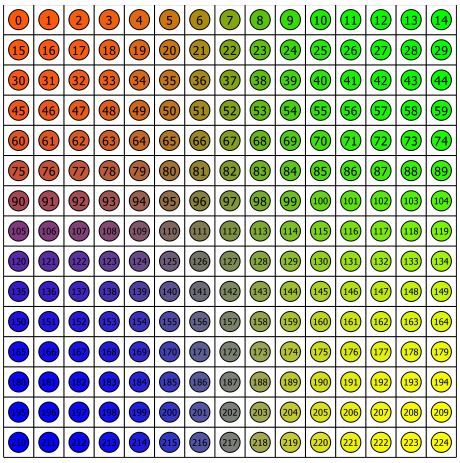
In a separate paper [1], we provide a number of breakthroughs for parallel motion planning.

- We show that it is strongly NP-complete to minimize the makespan for reconfiguring a system of labeled circular robots in a grid environment.
- We give an $\mathcal{O}(1)$-approximation for the long-standing open problems of parallel motion-planning with minimum makespan in a grid setting. This result is based on establishing an *absolute* performance guarantee: We prove that for any labeled arrangement of robots, there is always an overall schedule that gets each robot to its target destination with bounded *stretch*, i.e., within a constant factor of the largest individual distance.
- We extend our results to the scenario with continuous motion and arbitrary coordinates, provided the distance between a robot's start and target positions is at least one diameter. This implies that efficient multi-robot coordination is always possible under relatively mild separability conditions; this includes non-convex robots.
- For the continuous case with more densely packed objects, we establish a lower bound of $\Omega(N^{1/4})$ and an upper bound of $\mathcal{O}(\sqrt{N})$ on the achievable stretch.

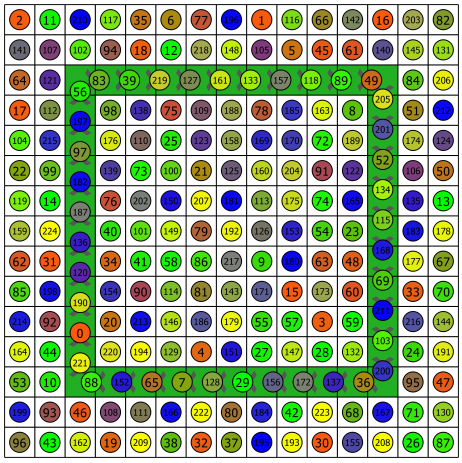
See [1] for technical aspects and a more detailed discussion of related work. The purpose of the video submitted with this abstract is to explain and visualize the theoretical approach.



(a) A start configuration.



(b) A target configuration.



(c) A feasible reconfiguration step.

Figure 1 The base scenario of N robots placed in a rectangular grid.

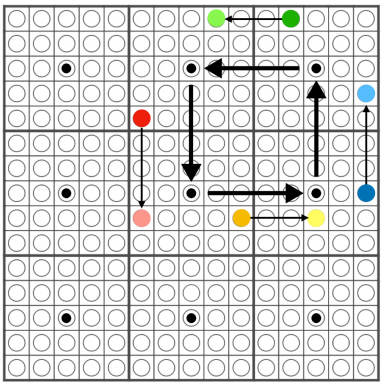
2 A base problem: labeled grid permutation

A fundamental scenario for multi-robot motion planning is the situation in which robots are placed at N grid positions (Fig. 1(a)); the desired target positions lie on the same grid, corresponding to a permutation of size N (Fig. 1(b)). In each step, a robot can move to an adjacent grid position if this is being vacated during the same step (Fig. 1(c)).

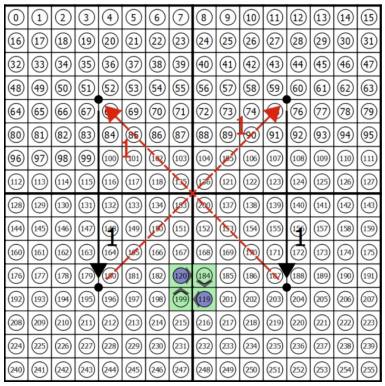
3 Complexity

We can show that even the base problem is NP-hard; see [1] for details.

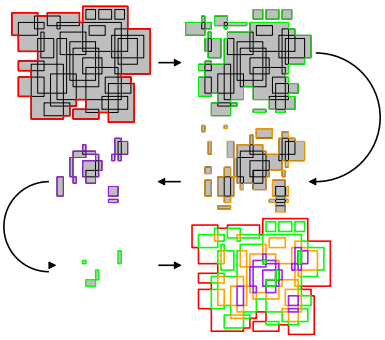
▶ **Theorem 1.** *The minimum makespan parallel motion planning problem on a grid is strongly NP-hard.*



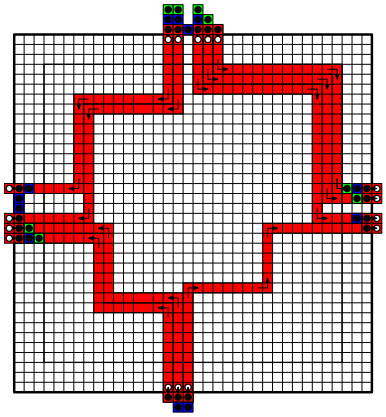
(a) Step 1: Computing the tiling T and the corresponding flow G_T.



(b) Step 2: Preprocessing G_T to remove intersecting and bidirectional edges.



(c) Step 3: Computing a partition into $\mathcal{O}(d)$ d-subflows.



(d) Step 4: Realizing the $\mathcal{O}(d)$ subflows using $\mathcal{O}(d)$ transformation steps.

Figure 2 The steps in Phase I.

4 Algorithmic description

The algorithm for achieving constant stretch in grid arrangements hinges on using local permutations for sorting the overall configuration. To achieve this in $\mathcal{O}(d)$ steps, the whole arrangement is subdivided into square tiles of size $\mathcal{O}(d) \times \mathcal{O}(d)$. In Phase I, all robots are moved to the tiles that contain their respective target positions, based on four steps that are based on flow techniques and illustrated in Fig. 2.

In Phase II, the disks within each tile are moved to their respective target positions in $\mathcal{O}(d)$ steps, based on local sorting methods. This results in an overall makespan of $\mathcal{O}(d)$.

5 Continuous motion

The result for the base case can be generalized to arbitrary geometric arrangements, provided that the start and target positions are sufficiently separated.

▶ **Theorem 2.** *If the distance between the centers of two robots of radius* 1 *in the start and target configurations is at least* $2\sqrt{2}$*, we can achieve a makespan in* $\mathcal{O}(d)$*, i.e., constant stretch, see Figure 3.*

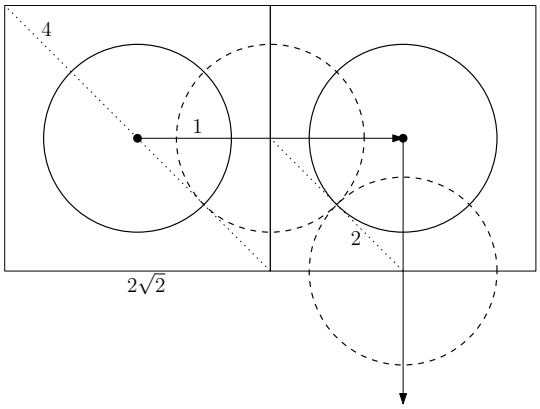
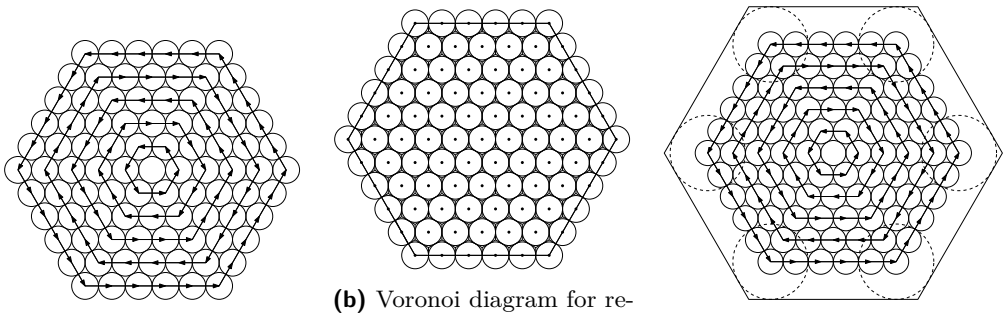


Figure 3 A mesh size of $2\sqrt{2}$ avoids robot collisions, and the cell diagonals have length 4. Note that robots may have arbitrary shape, as the separation argument applies to their circumcircles.



(a) Start and target positions of the robots.

(b) Voronoi diagram for restricted to the convex hull of the midpoints.

(c) Bounding polygon for the moving robots.

Figure 4 Lower-bound construction, with arrows pointing from start to target positions.

For more densely packed arrangements of disks, no constant stretch can be achieved.

▶ **Theorem 3.** *There is an instance with optimal makespan in $\Omega(N^{1/4})$, see Figure 4.*

Conversely, there is a non-trivial upper bound on the stretch, as follows.

▶ **Theorem 4.** *For a set of unit disks at arbitrary start and target positions there is a trajectory set with continuous makespan of $\mathcal{O}(d + \sqrt{N})$, implying a $\mathcal{O}(\sqrt{N})$-approximation algorithm.*

6 The video

The video opens with a number of practical illustrations for parallel motion planning, followed by a discussion of the work by Schwartz and Sharir, the underlying problem description and our complexity result. The main part focuses on the algorithm for achieving constant stretch, visualizing the individual steps of Phase I and Phase II. Finally, the basic aspects for continuous motion are sketched: achieving constant stretch for separable arrangements, as well as the lower and upper bounds for densely packed disks. In the very end, the geometric difficulty of finding optimal trajectories are shown, even for just two disks.

References

1 Erik D. Demaine, Sándor P. Fekete, Phillip Keldenich, Henk Meijer, and Christian Scheffer. Coordinated motion planning: Reconfiguring a swarm of labeled robots with bounded stretch. In Csaba Tóth and Bettina Speckmann, editors, *34th International Symposium on Computational Geometry (SoCG 2018, these proceedings)*, volume 99 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 29:1–29:17, 2018. Full version at `https://arxiv.org/abs/1801.01689`. `doi:10.4230/LIPIcs.SoCG.2018.29`.

2 Jacob T. Schwartz and Micha Sharir. On the piano movers' problem: III. Coordinating the motion of several independent bodies: the special case of circular bodies moving amidst polygonal barriers. *Int. J. Robotics Research*, 2(3):46–75, 1983.

Geometric Realizations of the 3D Associahedron

Satyan L. Devadoss
Department of Mathematics, University of San Diego
San Diego, CA 92129, USA
devadoss@sandiego.edu

Daniel D. Johnson
Department of Computer Science, Harvey Mudd College
Claremont, CA 91711, USA
ddjohnson@hmc.edu

Justin Lee
Department of Computer Science, Boston University
Boston, MA 02215, USA
jbwlee@bu.edu

Jackson Warley
Department of Computer Science, University of Massachusetts
Amherst, MA 01003, USA
jwarley@cs.umass.edu

Abstract

The associahedron is a convex polytope whose 1-skeleton is isomorphic to the flip graph of a convex polygon. There exists an elegant geometric realization of the associahedron, using the remarkable theory of secondary polytopes, based on the geometry of the underlying polygon. We present an interactive application that visualizes this correspondence in the 3D case.

2012 ACM Subject Classification Theory of computation → Computational geometry, Applied computing → Computer-assisted instruction

Keywords and phrases associahedron, secondary polytope, realization

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.75

Category Multimedia Exposition

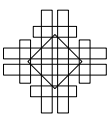
1 The associahedron

Let us begin with a formal definition of the associahedron; details are given in [13].

▶ **Theorem.** *Let $A(n)$ be the poset of sets of non-crossing diagonals of a convex polygon P with n vertices, ordered so that for any $a, a' \in A(n)$, we have $a \prec a'$ if a is obtained from a' by adding a new diagonal to P. The* associahedron $\mathcal{K}_{n-1}$ *is a simple convex polytope of dimension $n-3$ whose face poset is isomorphic to $A(n)$.*

By construction, each vertex of $\mathcal{K}_{n-1}$ corresponds to a triangulation of P, whereas each facet of $\mathcal{K}_{n-1}$ corresponds to a diagonal of P. Just as each triangulation T of P uses exactly $n-3$ diagonals, each vertex of $\mathcal{K}_{n-1}$ is incident to $n-3$ facets (making it a simple polytope), corresponding to the diagonals used in T. Furthermore, each edge of $\mathcal{K}_{n-1}$ connecting two vertices corresponds to an edge flip between the respective triangulations.

The associahedron was constructed independently by Haiman (unpublished) and Lee [13], though Stasheff had defined the underlying abstract object twenty years prior in his work on




associativity in homotopy theory [17]. The famous *Catalan numbers* enumerate the vertices, with over one hundred different combinatorial and geometric interpretations available. The beauty of this polytope is the multitude of settings in which it makes an appearance, including root systems [5], real algebraic geometry [6], computational geometry [8], phylogenetics [1], string theory [9], J-holomorphic curves [14], and hypergeometric functions [18].

Rather than just a combinatorial framework, $\mathcal{K}_n$ also has a rich *geometric* perspective. There are numerous realizations of the associahedron, obtained by taking the convex hull of the coordinates of its vertices. In a beautiful recent paper, Ceballos, Santos, and Ziegler classify all such realizations into three main groups [4]: The first one is based on *generalized permutohedra*, spearheaded by Postnikov, and constructs the associahedron from truncations of a simplex [16]. The second is based on *cluster complexes* of Coxeter type A_n, arising from the work of Fomin and Zelevinsky [5]. The final realization follows the classical work on *secondary polytopes* by Gelfand, Kapranov, and Zelevinsky [10].

The first two of these constructions have the property that $\mathcal{K}_n$ has exactly $n-2$ pairs of parallel facets. Indeed, Hohlweg and Lange [11] and Santos [4] showed that both of these realizations are particular cases of exponentially many different constructions of the associahedron. The third (secondary polytope) realization, denoted as GKZ, has a radically different flavor. It not only yields a polytope that never has parallel facets, but results in a continuous deformation of the associahedron obtained from a continuous deformation of its underlying polygon. We focus on an interactive visualization of this wonderful realization.

2 The GKZ realization

This realization provides a beautiful blend of combinatorics, geometry, and the convex hull; see [10] for the general theory and [7, Chapter 3] for a readable version. Let P be a planar convex geometric polygon with vertices $p_1, \ldots, p_n$. For a triangulation T of P, let

$$\psi_T(p_i) \;=\; \sum_{p_i \in \Delta \in T} \operatorname{area}(\Delta)$$

be the sum of the areas of all triangles Δ incident to the vertex p_i. In other words, the value $\psi_T(p_i)$ associated to each vertex p_i of the polygon is the sum of the areas of the triangles incident to p_i. Thus, to each triangulation of the convex polygon P, we obtain an *area vector*

$$\Phi(T) \;=\; (\psi_T(p_1), \ldots, \psi_T(p_n)) \in \mathbb{R}^n \,.$$

The number of potentially different area vectors associated to a given convex polygon with n vertices is the Catalan number. The following result comes from [10]:

▶ **Theorem.** *If P is a convex polygon with n vertices, the convex hull of the area vectors in $\mathbb{R}^n$ of all triangulations of P results in a geometric realization of the associahedron $\mathcal{K}_{n-1}$.*

What is remarkable about this result is that, although the convex hulls of the area vectors of different convex polygons are geometrically distinct, they output the same combinatorial structure for *any* convex polygon with n vertices.

3 Visualizing the 3D associahedron

We restrict our attention to $n = 6$, in which P is a planar convex hexagon with vertices $p_1, \ldots, p_6$. Thus, P has 14 distinct triangulations $T_1, \ldots, T_{14}$, and the theorem above ensures that the convex hull of the set of area vectors $\Phi(T_i)$ yields a realization $\mathfrak{K}$ of K_5 in $\mathbb{R}^6$. Since

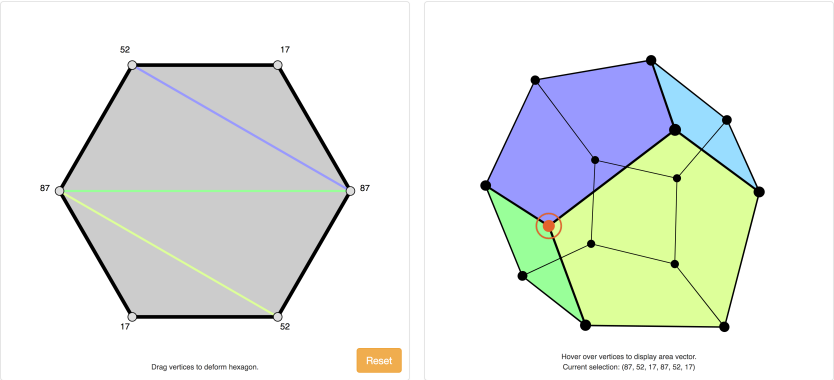


Figure 1 The interface of the web application. On the left, the convex hexagon, with one triangulation shown. On the right, the GKZ realization $\mathfrak{K}$ of $\mathcal{K}_5$, with the vertex corresponding to the triangulation highlighted in orange.

the GKZ realization lies in a 3-dimensional subspace $S \subseteq \mathbb{R}^6$, choosing an orthonormal basis $\mathcal{B}$ for S allows us to embed $\mathfrak{K}$ in $\mathbb{R}^3$ while preserving its geometric structure. We simply apply the Gram-Schmidt process to the vectors $\Phi(T_1), \Phi(T_2), \ldots, \Phi(T_{14})$, where the triangulations are ordered arbitrarily.

We provide an application that calculates the area vectors corresponding to each triangulation of P, and uses the Gram-Schmidt process to obtain an orthonormal basis for S. Our visualization application is implemented in HTML5 and JavaScript; it runs client-side in the web browser and can be accessed at `http://www.hexahedria.com/associahedron/`. Due to finite numerical precision, there may be small errors that cause our computed vertices of $\mathfrak{K}$ to lie slightly outside of the desired subspace S. To prevent this, we "snap" vectors with magnitude smaller than a numerical tolerance factor epsilon to zero, and project the vertices into the subspace spanned by the first three nonzero basis vectors. The program leverages several open source libraries: D3 [2] and Three.js [3] for graphics, numeric.js [15] for arithmetic, and PolyK [12] for simple polygon operations such as testing convexity.

The interface includes two side-by-side visuals: one of a triangulated convex hexagon P (left side), and one of the corresponding GKZ embedding $\mathfrak{K}$ (right side). The three diagonals of one triangulation T of P are displayed, and each vertex p_i of P is labeled with

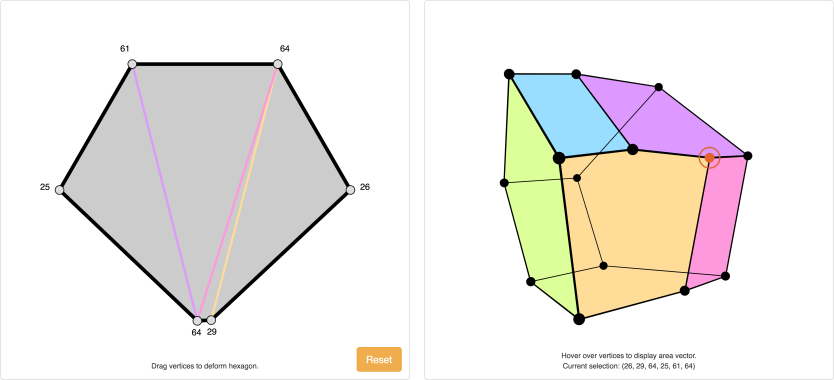


Figure 2 A deformed hexagon and its corresponding deformed associahedral realization.

the area $\psi_T(p_i)$ of the triangles incident to it in the triangulation. On the left side, users can deform the hexagon by manipulating its vertices. On the right side, users can rotate the associahedron and change the active triangulation by hovering over a different vertex. Moreover, each diagonal of the hexagon is color-coded to match the corresponding facet of the associahedron. As the user moves a vertex in the hexagon view, the areas of the triangles change and the polytope deforms accordingly. Indeed, under extreme deformations of the underlying polygon, highly interesting geometric representations appear, many in degenerate configurations.

Although our application does not allow nonconvex deformations of the underlying polygon, the mathematical theory is known [8]. The resulting "associahedral" structure will not be a convex polytope but a *polytopal complex*. This complex is not only (topologically) contractible, but possesses a geometric realization based on the theory of secondary polytopes.

References

1 Louis J Billera, Susan P Holmes, and Karen Vogtmann. Geometry of the space of phylogenetic trees. *Advances in Applied Mathematics*, 27(4):733–767, 2001.

2 Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D^3 data-driven documents. *IEEE transactions on visualization and computer graphics*, 17(12):2301–2309, 2011.

3 Ricardo Cabello et al. Three.js, 2010. URL: `https://github.com/mrdoob/three.js`.

4 Cesar Ceballos, Francisco Santos, and Günter M Ziegler. Many non-equivalent realizations of the associahedron. *Combinatorica*, 35:513–551, 2015.

5 Frederic Chapoton, Sergey Fomin, and Andrei Zelevinsky. Polytopal realizations of generalized associahedra. *Bulletin Canadien de Mathématiques*, 45:537–566, 2002.

6 Satyan L Devadoss. Tessellations of moduli spaces and the mosaic operad. *Contemporary Mathematics*, 239:91–114, 1999.

7 Satyan L Devadoss and Joseph O'Rourke. *Discrete and Computational Geometry*. Princeton University Press, 2011.

8 Satyan L Devadoss, Rahul Shah, Xuancheng Shao, and Ezra Winston. Deformations of associahedra and visibility graphs. *Contributions to Discrete Mathematics*, 7:68–81, 2012.

9 Kenji Fukaya, Yong-Geun Oh, Hiroshi Ohta, and Kaoru Ono. *Lagrangian intersection Floer theory: Anomaly and obstruction*. American Mathematical Society, 2010.

10 Israel M Gelfand, Mikhail Kapranov, and Andrei Zelevinsky. *Discriminants, Resultants, and Multidimensional Determinants*. Springer, 2008.

11 Christophe Hohlweg and Carsten Lange. Realizations of the associahedron and cyclohedron. *Discrete & Computational Geometry*, 37(4):517–543, 2007.

12 Ivan Kuckir. PolyK.js. URL: `http://polyk.ivank.net/`.

13 Carl W Lee. The associahedron and triangulations of the n-gon. *European Journal of Combinatorics*, 10:551–560, 1989.

14 Chiu-Chu Melissa Liu. Moduli of J-holomorphic curves with Lagrangian boundary conditions and open Gromov-Witten invariants for an S^1-equivariant pair. *arXiv math/0210257*, 2002.

15 Sébastien Loisel. Numeric javascript. URL: `http://www.numericjs.com/`.

16 Alexander Postnikov. Permutohedra, associahedra, and beyond. *International Mathematics Research Notices*, 2009:1026–1106, 2009.

17 Jim Stasheff. From operads to 'physically' inspired theories. In *Operads: Proceedings of Renaissance Conferences*, volume 202, page 53. American Mathematical Society, 1997.

18 M Yoshida. *Hypergeometric functions, my love*. Vieweg, 1997.

Star Unfolding of Boxes

Dani Demas
Technology, Amazon.com, Seattle, 98109 WA
daniellefdemas@gmail.com

Satyan L. Devadoss
Department of Mathematics, University of San Diego, San Diego, 92129 CA
devadoss@sandiego.edu

Yu Xuan Hong
Department of Mathematics, Pomona College, Claremont, 91711 CA
yu_xuan.hong@pomona.edu

Abstract

Given a convex polyhedron, the star unfolding of its surface is obtained by cutting along the shortest paths from a fixed source point to each of its vertices. We present an interactive application that visualizes the star unfolding of a box, such that its dimensions and source point locations can be continuously toggled by the user.

2012 ACM Subject Classification Mathematics of computing → Discrete mathematics, Theory of computation → Computational geometry

Keywords and phrases star unfolding, source unfolding, Voronoi diagram

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.76

Category Multimedia Exposition

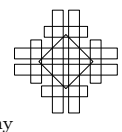
1 Star Unfolding

The problem of unfolding polyhedra was popularized by Albrecht Dürer in the early 16th century, when he cut polyhedra along edges and represented them in their unfolded state. This process of edge unfolding created *nets*: connected faces that lay flat in the plane without overlap. One of the most beautiful problems in computational geometry asks whether every convex polyhedron has a net. In the absence of a resolution to this problem, it is of interest to create nets under different rules, allowing the polyhedral surface cuts to be arbitrary rather than only along the edges. The *star unfolding* of Alexandrov [1] is one possible method, dating back to 1948.

▶ **Definition 1.** The *star unfolding* of a convex polyhedron P fixes a generic point x on P, and cuts along the shortest paths from x to every vertex of P.

A source point is *generic* when there is a unique shortest path to each vertex. The reason these cuts suffice to flatten polyhedron P into a polygon $U^*(P)$ is that all the points of curvature (at the vertices) are resolved by these cuts. It is by no means obvious that this resulting polygon $U^*(P)$ does not overlap, and is therefore a net; this was shown to be true in 1992 by Aronov and O'Rourke [2].

Note that if P has n vertices, then the star unfolding $U^*(P)$ is a polygonal net with $2n$ vertices: the n vertices from the polyhedron interleaved with n copies of the source point x. Figure 1 shows the star unfolding produced by cutting along eight paths of a box. Note that two edges of $U^*(P)$ incident to a vertex v of P have the same length — they are the two "sides" of the cut along the shortest path from x to v on P.



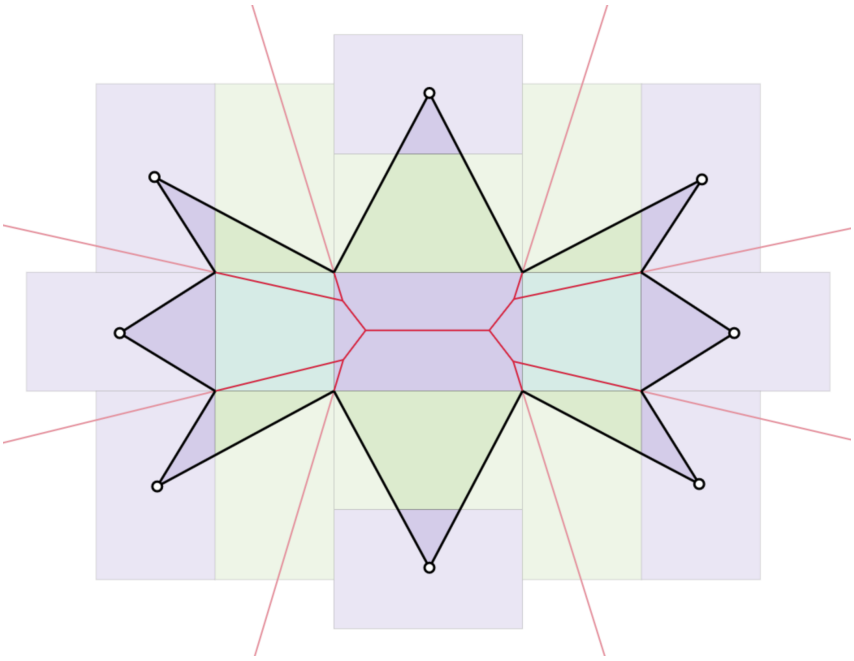



Figure 1 A star unfolding of a box.

Now we turn to another type of unfolding of a convex polyhedron, based on the cut locus.

▶ **Definition 2.** The *cut locus* $C(x)$ of a point x on the polyhedron is (the closure of) the set of all points y to which there is more than one shortest path from x. The *source unfolding* of a convex polyhedron P is obtained by cutting along $C(x)$ of a generic source point x.

The elegant underlying structure that relates the star and source unfoldings is the Voronoi diagram. In particular, the Voronoi diagram of the n copies of the source x, restricted to the interior of $U^*(P)$ (shown as red lines in Figure 1), is the cut locus $C(x)$ on P. Cutting along this Voronoi diagram in the star unfolding and rearranging the resulting n subpolygons yields the source unfolding. Indeed, the source and the star unfoldings are *scissors congruent* to one another [3, Chapter 1].

2 Algorithm for visualization

In our visualization, the user defines the dimensions of the box (up to scaling) along with the location of the source point. Pairs of opposite sides of the box are similarly color-coded, with the source point lying on a purple box face. Two steps are needed to obtain the star unfolding:

1. Find the shortest paths from the source point to each of the eight vertices of the box.
2. The relative order of the eight cuts is then used to determine the adjacency of edges in the star unfolding, allowing its construction.

Given a source point x, assume (without loss of generality) that it lies on the bottom face of the box. The shortest paths from x to the four vertices of the bottom face are simply straight line segments. To find the shortest paths from x to an arbitrary vertex v of the top face (as shown in Figure 2), there are six cases to consider: a path that goes through the bottom face and (1) the left face, (2) the back face, (3) the front and left faces, (4) the right

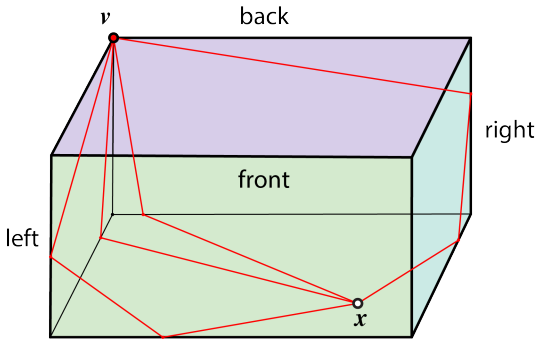


Figure 2 Four possible cases of shortest paths from source point x to vertex v.

and back faces, (5) the front and top faces, and (6) the right and top faces. However, since x is on the bottom face, it is straight forward to show that paths from (5) and (6) will never be shorter than those from (1) and (2), respectively. And so, there are only four cases that have a chance of yielding the shortest path to v, depending on the location of x. We calculate the shortest distance attainable in each of the four cases, and choose the path with minimal distance as a cut of the star unfolding.

Our visualization displays the background rectangles representing the faces that each cut passes through. The faces are unfolded onto the plane, so that each shortest cut is a straight line segment from a copy of the source point to a vertex; see Figure 3. Owing to the orthogonal nature of a box, adjacent edges that share a vertex are perpendicular to each other in the unfolding.

The Voronoi diagram of the eight copies of the source point is also drawn. Notice that when the source point crosses over the Voronoi diagram (with equidistant shortest paths to a vertex), the shape of the star unfolding will change, along with the background rectangles.

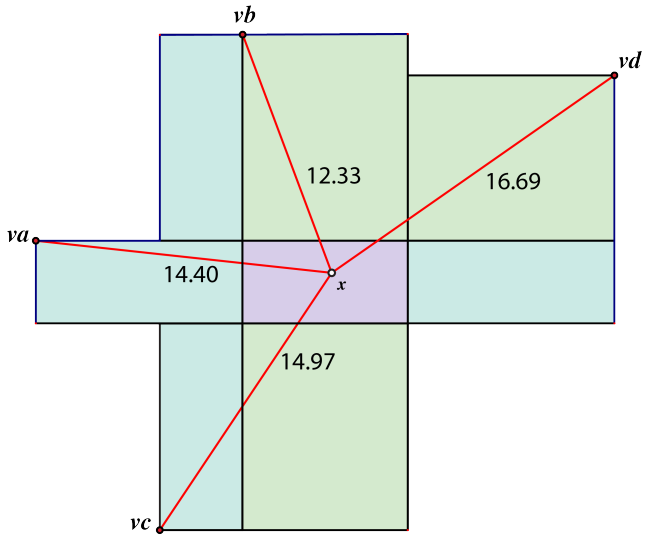


Figure 3 An example of possible cases of shortest path given source point x. The four points va, vb, vc, vd are copies of the same vertex v unfolded in ways corresponding to cases (1), (2), (3), and (4), respectively. In this example, the shortest path is to vertex copy vb.

3 Implementation

Our visualization is a browser application implemented in HTML5 and JavaScript, and can be accessed at `http://ddemas.github.io/box-unfolding-visualization/`. The user may adjust the *width*, *height*, and *length* of the box (up to rescaling) using the slide bars. The source point, located on the purple boxes, can also be altered, with dimensions *width* and *height*. The user also has the option to display or hide the Voronoi lines, background rectangles, and star unfolding.

The visualization works by constantly recalculating and redrawing the unfolding, using JQuery and Bootstrap libraries to help render the UI. The rectangle in the center and the four rectangles adjacent to it are fixed and represent five unique faces of the box. The rest of the rectangles change depending on the configuration that leads to the shortest path from the source point in the four corners to the nearest corner of the center rectangle, as explained in Section 2. The shortest distance to the remaining vertices of the box is inferred from the configuration of the rectangles in each corner.

Once the points of the box have been determined, we use an open source implementation of Fortune's algorithm written by Raymond Hill [4] to draw the Voronoi lines.

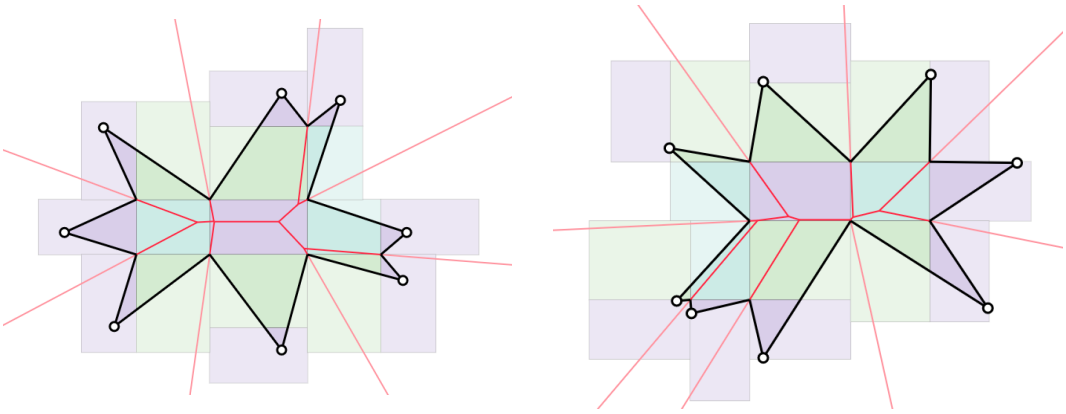


Figure 4 Screenshots from visualization that show examples from of different star unfoldings of the same box with different source points, with their underyling Voronoi diagrams.

References

1 A.D. Alexandrov. *Convex Polyhedra*. Springer Monographs in Mathematics. Springer Berlin Heidelberg, 2006. URL: `https://books.google.com/books?id=aoMreDT_DwcC`.

2 Boris Aronov and Joseph O'Rourke. Nonoverlap of the star unfolding. *Discrete & Computational Geometry*, 8(3):219–250, Sep 1992. `doi:10.1007/BF02293047`.

3 Satyan L. Devadoss and Joseph O'Rourke. *Discrete and Computational Geometry*. Princeton University Press, 2011.

4 Raymond Hill. Javascript-voronoi. `https://github.com/gorhill/Javascript-Voronoi`, 2013.

VoroCrust Illustrated: Theory and Challenges

Ahmed Abdelkader
University of Maryland, College Park MD, USA
akader@cs.umd.edu

Chandrajit L. Bajaj[1]
University of Texas, Austin TX, USA

Mohamed S. Ebeida
Sandia National Laboratories, Albuquerque NM, USA

Ahmed H. Mahmoud
University of California, Davis CA, USA

Scott A. Mitchell
Sandia National Laboratories, Albuquerque NM, USA

John D. Owens
University of California, Davis CA, USA

Ahmad A. Rushdi
University of California, Davis CA, USA

Abstract

Over the past decade, polyhedral meshing has been gaining popularity as a better alternative to tetrahedral meshing in certain applications. Within the class of polyhedral elements, Voronoi cells are particularly attractive thanks to their special geometric structure. What has been missing so far is a Voronoi mesher that is sufficiently robust to run automatically on complex models. In this video, we illustrate the main ideas behind the VoroCrust algorithm, highlighting both the theoretical guarantees and the practical challenges imposed by realistic inputs.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases sampling, surface reconstruction, polyhedral meshing, Voronoi

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.77

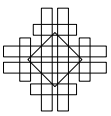
Category Multimedia Exposition

Related Version See the corresponding paper in these proceedings [2], `http://dx.doi.org/10.4230/LIPIcs.SoCG.2018.1`.

Funding This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research (ASCR), Applied Mathematics Program.

Acknowledgements Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

[1] Supported in part by a contract from Sandia, #1439100, and a grant from NIH, R01-GM117594



34th International Symposium on Computational Geometry (SoCG 2018).
Editors: Bettina Speckmann and Csaba D. Tóth; Article No. 77; pp. 77:1–77:4

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Mesh generation is a fundamental problem in computational geometry, geometric modeling, computer graphics, scientific computing and engineering simulations. There has been a growing interest in polyhedral meshes as an alternative to tetrahedral or hex-dominant meshes [17]. Polyhedra are less sensitive to stretching, which enables the representation of complex geometries without excessive refinement. In addition, polyhedral cells have more neighbors even at corners and boundaries, which offers better approximations of gradients and local flow distributions. Even compared to hexahedra, fewer polyhedral cells are needed to achieve a desired accuracy in certain applications. This can be very useful in several numerical methods [7]. In particular, the accuracy of a number of important solvers, e.g., the two-point flux approximation for conservation laws [14], greatly benefits from a conforming mesh which is *orthogonal* to its dual as naturally satisfied by Voronoi meshes. Such solvers play a crucial role in hydrology [19] and computational fluid dynamics [8].

A conforming volume mesh exhibits two desirable properties *simultaneously*: (1) a decomposition of the enclosed volume, and (2) a reconstruction of the bounding surface. Conforming Delaunay meshing is well-studied [11], but Voronoi meshing is less mature. A common practical approach to polyhedral meshing is to dualize a tetrahedral mesh and *clip* each cell by the bounding surface [15]. Unfortunately, clipping does not guarantee the convexity and connectedness of cells [13]. Another approach is to *mirror* Voronoi sites across the surface [20], but we are not aware of any approximation guarantees in this category.

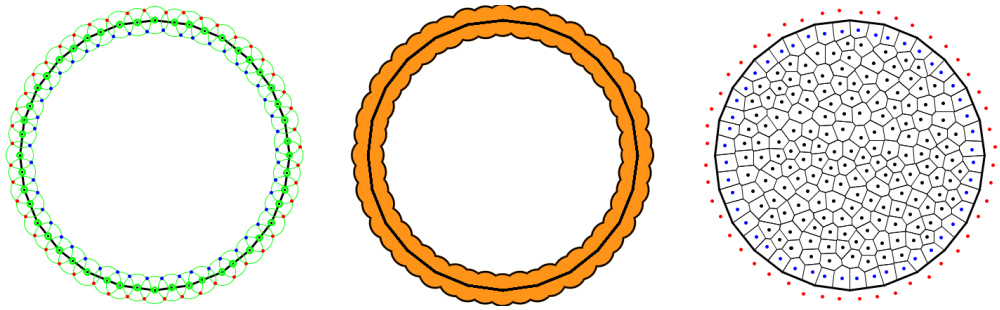
The desired mesher takes as input a description of the geometric model for which a conforming Voronoi mesh is to be generated, e.g., a set of surface samples or a complete computer-aided design (CAD) model. Having access to a CAD model allows the mesher to accurately sample new points as needed; a crucial requirement for high quality meshing. Deferring the sampling problem, we start in an idealized setting that allows us to establish strong theoretical guarantees on the correctness of the underlying approach. In a forthcoming paper [3], we describe the design and implementation of the complete VoroCrust algorithm, which can handle realistic inputs possibly having sharp features, can estimate a sizing function and generate samples, and can guarantee the quality of the output mesh.

In this multimedia contribution, we illustrate the basic constructions underlying the proposed algorithm and briefly explain why it works. A number of meshes generated by a preliminary implementation is shown. Finally, we highlight the main technical challenges imposed by realistic inputs along with an illustration of each. The rest of this abstract follows the same structure and concludes by a summary of the methods used in preparing the video content.

2 The abstract algorithm

An abstract version of the proposed algorithm, geared towards volumes bounded by smooth surfaces, can be described as follows. The points used in defining the Voronoi diagram are referred to as *seeds*. Figure 1 illustrates the steps in 2D.

1. Take as input a sample $\mathcal{P}$ on the surface $\mathcal{M}$ bounding the volume $\mathcal{O}$.
2. Define a ball B_i centered at each sample p_i, with a suitable radius r_i, and let $\mathcal{U} = \cup_i B_i$.
3. Initialize the set of seeds $\mathcal{S}$ with the corners of $\partial\mathcal{U}$, $\mathcal{S}^{\uparrow}$ and $\mathcal{S}^{\downarrow}$, on both sides of $\mathcal{M}$.
4. Optionally, generate additional seeds $\mathcal{S}^{\downarrow\downarrow}$ in the interior of $\mathcal{O}$, and include $\mathcal{S}^{\downarrow\downarrow}$ into $\mathcal{S}$.
5. Compute the Voronoi diagram $\mathrm{Vor}(\mathcal{S})$ and retain the cells with seeds in $\mathcal{S}^{\downarrow} \cup \mathcal{S}^{\downarrow\downarrow}$ as the volume mesh $\hat{\mathcal{O}}$, where the facets between $\mathcal{S}^{\uparrow}$ and $\mathcal{S}^{\downarrow}$ yield a surface approximation $\hat{\mathcal{M}}$.



(a) Seeds from union of balls. **(b)** Isotopic reconstruction. **(c)** Conforming decomposition.

Figure 1 Screen captures from the video illustrating the algorithmic steps on a planar curve.

Under appropriate conditions on the sampling, the union of balls U is isotopic to the bounding surface $\mathcal{M}$ [10]. In addition, the Voronoi facets in $\mathrm{Vor}(\mathcal{S})$ separating $\mathcal{S}^{\uparrow}$ and $\mathcal{S}^{\downarrow}$, which constitute the surface reconstruction $\hat{\mathcal{M}}$, are in fact the medial axis of U [6], which is isotopic to $\mathcal{M}$ if the balls have suitable spacing and radii [9]. The sparsity condition helps in bounding the quality of mesh elements; the cells are fat. Finally, the enclosed volume can easily be refined without disrupting the surface reconstruction.

In particular, we assume $\mathcal{P}$ is an ϵ-sample with a weak σ-sparsity condition dictating $\mathbf{d}(p_i, p_j) \geq \sigma\epsilon \cdot \mathrm{lfs}(p_j)$ whenever $\mathrm{lfs}(p_i) \geq \mathrm{lfs}(p_j)$. The union of balls $\mathcal{U}$ is defined by setting $r_i = \delta \cdot \mathrm{lfs}(p_i)$. Fixing $\delta = 2\epsilon$, the sampling conditions for non-uniform approximation are satisfied when ϵ is sufficiently small [10]. We also fix $\sigma = 3/4$ to give $\mathcal{U}$ a particularly nice structure which guarantees all samples appear as vertices in the surface reconstruction.

In this setting, we establish a number of geometric properties of the construction. This allows us to bound the distance between seeds in $\mathcal{S}^{\uparrow} \cup \mathcal{S}^{\downarrow}$ and their separating facets, which in turns bounds the inradius of Voronoi cells in the resulting decomposition $\hat{\mathcal{O}}$. To bound the outradius of cells, we proceed to seed the interior of $\mathcal{O}$. Assuming $\mathcal{O}$ is shifted and scaled to fit in the root box of the octree, we refine octree boxes until their sizes match an extension of lfs. Once refinement terminates, we insert into $\mathcal{S}^{\downarrow\downarrow}$ additional seeds at the centers of all boxes which are empty of seeds in $\mathcal{S}^{\uparrow} \cup \mathcal{S}^{\downarrow}$. This scheme allows us to bound the ratio of outradius to inradius for all cells, and also to bound the number of seeds. See [2] for the details and [4] for some earlier related work using a different scheme.

3 Dealing with realistic inputs

Unlike the idealized setting studied above, realistic inputs pose a number of challenges. Namely, sharp features do not fit well with the ϵ-sampling paradigm and require extra care to bound the quality of mesh elements in their neighborhoods. Even without sharp features, estimating the local feature size is impractical. In addition, for applications requiring surface approximations with good normals, extra steps are needed to iron out any irregularities that the basic approach might yield. We address these issues in a forthcoming publication [3].

4 Video production

We used *ParaView* [5] and the *Processing* programming language [18] to generate our animations. Speech was synthesized via the *Bing Speech API* [12] and the video was composed using *OpenShot* [16]. A low-quality version of the video will be hosted on the conference website [1].

References

1. A. Abdelkader, C. Bajaj, M. Ebeida, A. Mahmoud, S. Mitchell, J. Owens, and A. Rushdi. VoroCrust illustrated: theory and challenges (video content). http://computational-geometry.org/SoCG-videos/socg18video/videos/VoroCrust/.
2. A. Abdelkader, C. Bajaj, M. Ebeida, A. Mahmoud, S. Mitchell, J. Owens, and A. Rushdi. Sampling conditions for conforming Voronoi meshing by the VoroCrust algorithm. In *34th International Symposium on Computational Geometry (SoCG 2018)*, pages 1:1–1:16, 2018. doi:10.4230/LIPIcs.SoCG.2018.1.
3. A. Abdelkader, C. Bajaj, M. Ebeida, A. Mahmoud, S. Mitchell, J. Owens, and A. Rushdi. VoroCrust: Voronoi meshing without clipping. *Manuscript*, In preparation.
4. A. Abdelkader, C. Bajaj, M. Ebeida, and S. Mitchell. A Seed Placement Strategy for Conforming Voronoi Meshing. In *Canadian Conference on Computational Geometry*, 2017.
5. J. Ahrens, B. Geveci, and C. Law. Paraview: An end-user tool for large-data visualization. In C. Hansen and C. Johnson, editors, *Visualization Handbook*, pages 717–731. Butterworth-Heinemann, 2005.
6. N. Amenta and R.-K. Kolluri. The medial axis of a union of balls. *Computational Geometry*, 20(1):25–37, 2001. Selected papers from the 12th Annual Canadian Conference.
7. N. Bellomo, F. Brezzi, and G. Manzini. Recent techniques for PDE discretizations on polyhedral meshes. *Mathematical Models and Methods in Applied Sciences*, 24(08):1453–1455, 2014.
8. T. Brochu, C. Batty, and R. Bridson. Matching fluid simulation elements to surface geometry and topology. *ACM Trans. Graph.*, 29(4):47:1–47:9, 2010.
9. F. Chazal and D. Cohen-Steiner. A condition for isotopic approximation. *Graphical Models*, 67(5):390–404, 2005. Solid Modeling and Applications.
10. F. Chazal and A. Lieutier. Smooth manifold reconstruction from noisy and non-uniform approximation with guarantees. *Computational Geometry*, 40(2):156–170, 2008.
11. S.-W. Cheng, T. Dey, and J. Shewchuk. *Delaunay Mesh Generation*. CRC Press, 2012.
12. Microsoft Corporation. Bing Speech API. https://azure.microsoft.com/en-us/services/cognitive-services/speech/.
13. M. Ebeida and S. Mitchell. Uniform random Voronoi meshes. In *International Meshing Roundtable (IMR)*, pages 258–275, 2011.
14. R. Eymard, T. Gallouët, and R. Herbin. Finite volume methods. In *Techniques of Scientific Computing (Part 3)*, volume 7 of *Handbook of Numerical Analysis*, pages 713–1018. Elsevier, 2000.
15. R.-V. Garimella, J. Kim, and M. Berndt. Polyhedral mesh generation and optimization for non-manifold domains. In *International Meshing Roundtable (IMR)*, pages 313–330. Springer International Publishing, 2014.
16. OpenShot Studios, LLC. OpenShot Video Editor. http://www.openshot.org/.
17. M. Peric and S. Ferguson. The advantage of polyhedral meshes. *Dynamics - Issue 24*, page 4–5, Spring 2005. The customer magazine of the CD-adapco Group, currently maintained by Siemens at http://siemens.com/mdx. The issue is available at http://mdx2.plm.automation.siemens.com/magazine/dynamics-24 (accessed March 29, 2018).
18. C. Reas and B. Fry. *Processing: A Programming Handbook for Visual Designers and Artists*. The MIT Press, 2014.
19. M. Sents and C. Gable. Coupling LaGrit Unstructured Mesh Generation and Model Setup with TOUGH2 Flow and Transport. *Comput. Geosci.*, 108(C):42–49, 2017.
20. M. Yip, J. Mohle, and J. Bolander. Automated modeling of three-dimensional structural components using irregular lattices. *Computer-Aided Civil and Infrastructure Engineering*, 20(6):393–407, 2005.